2015

Adaptive Control For Autonomous Navigation Of Mobile Robots Considering Time Delay And Uncertainty

Stephen Kofi Armah
North Carolina Agricultural and Technical State University

Adaptive Control for Autonomous Navigation of Mobile Robots Considering Time Delay and Uncertainty

Stephen Kofi Armah

North Carolina A&T State University

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department: Mechanical Engineering

Major: Mechanical Engineering

Major Professor: Dr. Sun Yi

Greensboro, North Carolina

2015

The Graduate School
North Carolina Agricultural and Technical State University

This is to certify that the Doctoral Dissertation of

Stephen Kofi Armah

has met the dissertation requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2015

Approved by:

Dr. Sun Yi
Major Professor

Dr. Samuel Owusu-Ofori
Committee Member

Dr. Jung H. Kim
Committee Member

Dr. Frederick Ferguson
Committee Member

Dr. Mannur Sundaresan
Committee Member

Dr. Samuel Owusu-Ofori
Department Chair

Dr. Sanjiv Sarin
Dean, The Graduate School

Biographical Sketch

Stephen Kofi Armah was born in Accra, Ghana. He earned his Bachelor of Science degree in Mechanical Engineering from the Kwame Nkrumah University of Science and Technology (KNUST), Kumasi, Ghana in 1999. He received a Master of Science degree in Advanced Mechanical Engineering from the Imperial College of Science, Technology and Medicine, London, UK in 2003. He later received a Postgraduate Certificate in Education from The Open University, Manchester, UK in 2009. Stephen joined the Doctor of Philosophy program of the Mechanical Engineering Department at NC A&T State University in the fall of 2011.

Stephen has received various honors and recognition at all levels of his academic carrier, which include: Certificate of Academic Honors at NC A&T State University (2012-2015), Excellence in Academic Performance at The Open University (2009), Best Mechanical Engineering Student at KNUST (1999), Shell and Poku Transport award at KNUST (1995-1999), Co-founder NSBE-Ghana (2000), and Best Student at his High School's District (1991).

His current research on autonomous mobile robot controls, supervised by Dr. Sun Yi, has been presented at conferences and submitted to journal papers, including: ASME District F conference (Birmingham, Alabama, 2013), American Journal of Engineering and Applied Sciences (2014), Advances in Mechanical Engineering (2015), IFAC TDS Workshop (Ann Arbor, Michigan, 2015), and International Journal of Control, Automation and Systems (2015). His previous research work include Stress Analysis on Artificial Elbow Joints Using Finite Element Analysis and Computer Controlled Maintenance Schedule.

Stephen has over fifteen years of experience in teaching and learning in high school and college education in Ghana, UK, and US. He has also carried out assessment work for Edexcel, OCR, and AQA in UK for the past ten years.

Dedication

I dedicate this Doctoral Dissertation to almighty God, who has been my guidance throughout my entire educational career. I also dedicate this work to my wonderful mother, Mary Ama Nyamekye. Without her love and hardwork, I would not have been able to reach this goal.

Acknowledgments

I acknowledge and express great gratitude toward Dr. Sun Yi for his professional advice, critics, and important suggestions at different stages of this research work as well as for the editing of this dissertation.

Also, I am very thankful to my doctoral committee members Dr. Samuel Owusu-Ofori, Dr. Frederick Ferguson, Dr. Jung H. Kim, and Dr. Mannur Sundaresan for their advice and important suggestions of this research work.

Furthermore, I would also like to extend my sincerest thanks and appreciation to the NC A&T graduate school representative Dr. Musibau Shofoluwe.

Last but not least, I am very grateful to Deirdre Corbett, Tajanae Barnes, and Sonny Goodson of the NC A&T State University Writing Center for their editing of this dissertation.

Table of Contents

List of Figures

List of Tables

Abbreviations

AC	Adaptive Control
AO	Obstacle Avoidance
API	Application Program Interface
ARX	AutoRegressive eXogenous
AUV	Autonomous Underwater Vehicle
DDE	Delay Differential Equation
DDWMR	Differential Drive Wheeled Mobile Robot
FSF	Full-state Feedback
FSM	Finite State Machine
FW	Forward-wall
GTG	Go-to-goal
GUI	Graphical User Interface
HOS	Higher-order Statistics
HPF	High Pass Filter
LQR	Linear Quadratic Regulator
LTI	Linear Time-invariant
MPC	Model Predictive Control
MRAC	Model Reference Adaptive Control
ODE	Ordinary Differential Equation
P	Proportional
PD	Proportional-derivative
PDF	Probability Density Function

PID	Proportional-integral-derivative
PV	Proportional plus Velocity
RC	Robust Control
RMS	Root Mean Square
SDK	Software Development Kit
SISO	Single-input-single-output
TDE	Time-delay Estimation
TDOA	Time Delay of Arrival
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VTOL	Vertical Take-off and Landing
WMR	Wheeled Mobile Robot

Physical Constants

Constant Name	Symbol = Value (with units)
AR.Drone 2.0 Flight Management System Sampling Time	$T_s = 0.065$ s
AR.Drone 2.0 Overall Mass (with Indoor Hull)	$m_{ar} = 0.453$ kg
Distance between Quadrotor's Rotor and Center of Mass	$d_q = 0.2319$ m
Gravitational Strength	$g = 9.81$ ms^{-2}
Quadrotor Moments of Inertial about the x and y Axes	$J_{xx} = J_{yy} = 0.0125$ kgm^2
Quadrotor Moment of Inertial about the z Axis	$J_{zz} = 0.0287$ kgm^2
Quadrotor's Total Mass	$m_q = 1.5$ kg
Thrust Constant	$a = 4.3248$ x 10^{-5} Ns^2
Torque Constant	$b = 5.9693$ x 10^{-8} Nms^2

Symbols

Symbol	MATLAB/Simulink	Description	Units
{V}	–	Vehicle Frame/Coordinate System	–
{B}	–	Body Frame/Coordinate System	–
{O}/{I}	–	Global Inertia Frame/Coordinate System	–
A	A	System (State) Matrix	–
B	B	Input Matrix	–
C	C	Output Matrix	–
D	D	Feedforward/Feedthrough Matrix	–
s	s	Laplace Transform Operator	–
z	z	Z Transform Operator	–
$\Re$	–	Real Plane	–
$\mathbb{C}$	–	Complex Plane	–
$CONT$	cont	Controllability Matrix	–
$OBSER$	obser	Observability Matrix	–
t	t	Time	s
x	x	State Vector	–
y	y	Output Vector	–
u	u	Control Input Vector	–
$\mathbf{u}_m$	um	Plan Model Control Input Vector	–
e	e	Error	–
r	r	Reference Input	–
K_P	Kp	Proportional Gain	–
K_I	Ki	Integral Gain	–
K_D	Kd	Derivative Gain	–
ξ	zeta	Plant/Model Damping Ratio	–
ξ_f	zeta_f	Filter Damping Ratio	–
ω_n	wn	Plant/Model Natural Frequency	rads^{-1}
ω_f	wf	Filter Natural Frequency	rads^{-1}
K	K	Control Gains Matrix	–

K	K	Real Constant Tuning Parameter	–
$\mathbf{I}$	I	Identity Matrix	–
I	I	Desired Closed-loop Poles (DCLP)	–
β_1	beta_1	Real Part of DCLP	–
β_2	beta_2	Imaginary Part of DCLP	–
J_c	–	Cost Function	–
J_r	–	Rotational Inertia	kgm^2
$\mathbf{J}$	–	Rotational Inertia Matrix	kgm^2
B_r	–	Aerodynamic Damping	–
G_p	Gp	Plant Transfer Function	–
G_r	–	Plant Transfer Function without Time Delay	–
G_m	Gm	Reference Model Transfer Function	–
G_{mt}	–	DC Motor Dynamics Transfer Function	–
$\boldsymbol{\theta}$	theta	MIT Rule MRAC Updating Parameter	–
γ	gamma	MRAC Tuning Parameter	–
T_d	Td	Time Delay	s
Δ_m	delta_m	Added/Perturbed Mass	kg
m_q	m	Quadrotor's Total Mass	kg
m_{ar}	mr	AR.Drone 2.0 Overall Mass (Indoor)	kg
m_e	me	Mass at End of Cross Arm of Quadrotor	kg
m_c	mc	Mass of Quadrotor Center	kg
T_s	Ts	AR.Drone 2.0 System Sampling Time	s
d_q	d	Distance between Quadrotor's Rotor and Center of Mass	m
$\omega_B/\omega_{B/I}$	wB	Quadrotor's Angular Velocity w.r.t. {B}	rads^{-1}
${}^{B}\mathbf{R}_V$	RT	Rotational Matrix from {V} to {B}	–
$\mathbf{v}_B$	–	Quadrotor's Linear Velocity w.r.t. {B}	ms^{-1}
$\mathbf{F}_B$	–	Total Force on Quadrotor w.r.t. {B}	N
$\boldsymbol{\Gamma}_B$	–	Applied Torque on Quadrotor w.r.t. {B}	Nm

$\mathbf{h}_B$	–	Quadrotor Angular Momentum w.r.t. {B}	$\mathrm{kgm^2s^{-1}}$
$\omega_i, i = 1,2,3,4$	wi	Quadrotor Rotors Angular Velocity	$\mathrm{rads^{-1}}$
$T_i, i = 1,2,3,4$	Ti	Quadrotor Rotors Upward Thrust	N
$\tau_i, i = 1,2,3,4$	τi	Aerodynamic Torque on Quadrotor Rotors	Nm
T	T	Quadrotor Total Upward Thrust	N
g	g	Gravitational Strength	$\mathrm{ms^{-2}}$
(J_{xx}, J_{yy}, J_{zz})	(Jx, Jy, Jz)	Quadrotor Moments of Inertial About the x, y and z axes	$\mathrm{kgm^2}$
a	a	Thrust Constant	$\mathrm{Ns^2}$
b	b	Torque Constant	$\mathrm{Nms^2}$
(x, y, z)	–	DDWMR/Quadrotor Position Coordinates	m
(ρ, α, β)	–	Polar Coordinates	rad
x_g	xg	Desired x-value	m
y_g	yg	Desired y-value	m
$\mathbf{x}_g$	xg	Desired Goal State	m
$\mathbf{x}_o$	xo	Initial State	m
$\mathbf{x}_f$	xf	Final State	m
φ_g	yaw	Desired Heading/Yaw/Turn	rad or ^{o}
φ_{obs}	yawObs	Obstacle Heading/Yaw/Turn	rad or ^{o}
$\emptyset$	roll	Roll	rad or ^{o}
θ	pitch	Pitch	rad or ^{o}
φ	yaw	Heading/Yaw/Turn	rad or ^{o}
$(\emptyset, \theta, \varphi)$	–	Quadrotor Rotational Coordinates	rad or ^{o}
ω	w	Heading/Yaw/Turn Rate	$\mathrm{rads^{-1}}$

ω_r	wr	Right Angular Velocity	rads^{-1}
ω_l	wl	Left Angular Velocity	rads^{-1}
$k_v, k_\alpha, k_\beta, k_\rho, k_h, k_d$	Kv, Kalpha, Kbeta, Krho, Kh, Kd	Control Gains	–
R	R	Radius of the DDWMR Wheels	m
L	L	Distance the Between Wheels of the DDWMR	m
v	v	Linear Velocity	ms^{-1}
v_o	vo	Constant Linear Velocity	ms^{-1}
τ_s	tau	Time at Last Switch	s
$\tau(s)$	–	Torque Input Signal	Nm
d_{τ_s}	dtau	Distance Between Goal and Point at Last Switch	m
α_B	alphaBlend	Blending Function	–
β_B	betaBlend	Tuning Parameter for the Blending Function	–
u_{GTG}	–	Go-to-goal Mode	–
u_{AO}	–	Obstacle Avoidance Mode	–
u_{FW}^{c}	–	Clockwise Follow-wall Mode	–
u_{FW}^{cc}	–	Counterclockwise Follow-wall Mode	–
Δ_o	delta	Distance Between Robot and Obstacle	m
Δ_x/Δ_y	–	Change in x/y	m
e'	–	Corrected Error	rad
K_g	Kg	Quadrotor DC Motor Dynamics Steady-State Gain	$\text{rads}^{-1}\text{V}^{-1}$

τ	tau	Quadrotor DC Motor Dynamics Time Constant	s
$r_{\Re}$	rR	Real Stability Radius	–
Δ	delta	Perturbation Matrix	–
E and **F**	E and F	Scaling Matrices for Δ	–
Δ_{k_g}	delta_K	Perturbation in Quadrotor DC Motor Dynamics Steady-State Gain	$\mathrm{rads}^{-1}\mathrm{V}^{-1}$
Δ_{τ_g}	delta_tau	Perturbation in Quadrotor DC Motor Dynamics Time Constant	s

Abstract

Autonomous control of mobile robots has attracted considerable attention of researchers in the areas of robotics and autonomous systems during the past decades. One of the goals in the field of mobile robotics is development of platforms that robustly operate in given, partially unknown, or unpredictable environments and offer desired services to humans. Autonomous mobile robots need to be equipped with effective, robust and/or adaptive, navigation control systems. In spite of enormous reported work on autonomous navigation control systems for mobile robots, achieving the goal above is still an open problem. Robustness and reliability of the controlled system can always be improved.

The fundamental issues affecting the stability of the control systems include the undesired nonlinear effects introduced by actuator saturation, time delay in the controlled system, and uncertainty in the model. This research work develops robustly stabilizing control systems by investigating and addressing such nonlinear effects through analytical, simulations, and experiments. The control systems are designed to meet specified transient and steady-state specifications. The systems used for this research are ground (Dr Robot X80SV) and aerial (Parrot AR.Drone 2.0) mobile robots.

Firstly, an effective autonomous navigation control system is developed for X80SV using logic control by combining ‘go-to-goal’, ‘avoid-obstacle’, and ‘follow-wall’ controllers. A MATLAB robot simulator is developed to implement this control algorithm and experiments are conducted in a typical office environment. The next stage of the research develops an autonomous position (x, y, and z) and attitude (roll, pitch, and yaw) controllers for a quadrotor, and PD-feedback control is used to achieve stabilization. The quadrotor’s nonlinear dynamics and kinematics are implemented using MATLAB S-function to generate the state output.

Secondly, the white-box and black-box approaches are used to obtain a linearized second-order altitude models for the quadrotor, AR.Drone 2.0. Proportional (P), pole placement or proportional plus velocity (PV), linear quadratic regulator (LQR), and model reference adaptive control (MRAC) controllers are designed and validated through simulations using MATLAB/Simulink. Control input saturation and time delay in the controlled systems are also studied. MATLAB graphical user interface (GUI) and Simulink programs are developed to implement the controllers on the drone.

Thirdly, the time delay in the drone's control system is estimated using analytical and experimental methods. In the experimental approach, the transient properties of the experimental altitude responses are compared to those of simulated responses. The analytical approach makes use of the Lambert W function to obtain analytical solutions of scalar first-order delay differential equations (DDEs). A time-delayed P-feedback control system (retarded type) is used in estimating the time delay. Then an improved system performance is obtained by incorporating the estimated time delay in the design of the PV control system (neutral type) and PV-MRAC control system.

Furthermore, the stability of a parametric perturbed linear time-invariant (LTI) retarded-type system is studied. This is done by analytically calculating the stability radius of the system. Simulation of the control system is conducted to confirm the stability. This robust control design and uncertainty analysis are conducted for first-order and second-order quadrotor models.

Lastly, the robustly designed PV and PV-MRAC control systems are used to autonomously track multiple waypoints. Also, the robustness of the PV-MRAC controller is tested against a baseline PV controller using the payload capability of the drone. It is shown that the PV-MRAC offers several benefits over the fixed-gain approach of the PV controller. The adaptive control is found to offer enhanced robustness to the payload fluctuations.

CHAPTER 1

Introduction

This chapter discusses the motivation and problem statement for this research. This is followed by describing the two systems used for the research, and also reporting the major work done in this research area. It then continues with presenting the significance, objectives, and contributions of the research work. The chapter ends with presenting the approaches applied to address the problems and the research scope.

1.1 Motivation and Problem Statement

1.1.1 Motivation. A mobile robot is an automatic machine that is capable of movement in a desired way and there exist various types. Autonomous control of mobile robots has been a focus of active research in the past decades, and almost every major university has one or more labs for mobile robot research [15]. Autonomous mobile robots need to be equipped with effective, robust and/or adaptive, navigation control systems.

An autonomous mobile robot intended to perform its functions must be able to generate trajectories that are safe, smooth, and comfortable. Cutting-edge research is continuing to increase the viability of aerial robots (quadcopters) by making advances in multi-craft communication, environment exploration, and maneuverability [16]. If all of these developing qualities are combined together, quadcopters can become capable of advanced autonomous missions currently not possible with any other vehicle [16].

Mobile robotics is one of the fastest growing fields of engineering. According to International Federation of Robotics (IFR), in 2013, about 4 million service robots for personal and domestic use were sold, 28% more than in 2012; the value of sales increased to US$1.7 billion [17]. Cutting-edge sensor technologies such as high-definition light detection and ranging

(LIDAR) and stereo vision, in addition to the evolution of robotics architectures and development tools, are allowing these complex devices to become increasingly common [18]. By the end of 2015, the US Department of Defense (DOD) has mandated that one third of all military vehicles must be autonomous [18]. With the US Department of Transportation's Federal Aviation Administration (FAA) release of proposed framework regulations for unmanned aircraft systems, commercial drones, the future of autonomous aerial mobile is coming fast. The CEO of 3D Robotics, Chris Anderson, speaking at an event on the future vision of the world, said [19]:

> *Drones will collect detailed data for industrial applications and fly above an individual's head like a pet bird. Now that we've made these things work, it's time to work with them. The applications are huge, it's going to be a huge effort to integrate aerial robotics into the world of agricultural scenery and technology. 3D Robotics wants to be ready when it does.*

1.1.2 Problem statement. Mobile robot control can be seen as a mixture of engineering and cognitive science, and as such it presents unusual concerns to the control systems engineer [20]. One of the goals in the field of mobile robotics is development of platforms that robustly operate in given, partially unknown, or unpredictable environments and offer desired services to humans. In order to achieve this goal, autonomous mobile robots need to be equipped with appropriate control systems. Such control systems are supposed to have navigation control algorithms that will make mobile robots successfully accomplish different motions and behaviors.

In spite of enormous reported work on autonomous navigation systems for mobile robots using different algorithms such as proportional-integral-derivative (PID), pole placement, model predictive control (MPC), linear quadratic Gaussian (LQG), adaptive control (AC), artificial

intelligent (AI), machine learning, and fuzzy logic, achieving the goal above is still an open problem. Robustness and reliability of the controlled system can always be improved.

Firstly, actuator saturation is one issue that needs to be considered when designing control systems. Secondly, the time delay in the controlled system have to be investigated and possibly estimated, and then taken into account in the design. Moreover, the system to be controlled has uncertainty, internally and externally, and needs to be considered in designing the control systems. These undesired nonlinearities, among others, affects the stability and performance of the control systems.

Without an effective, adaptive and/or robust control algorithms to deal with such nonlinear effects, stabilization of autonomous navigation control systems for mobile robots remain a problem. This research work develops robustly stabilized control systems by investigating and addressing such nonlinear effects through analytical, simulations, and experiments.

1.2 Significance of Work

The main highlight of this research work is the development of an effective adaptive control system to address the parametric uncertainty in processes or systems such as an aircraft. The robustly designed control system has been successfully implemented to deal with payload fluctuations of quadcopter. The results obtained shows the benefits adaptive control offers over fixed-gain approach. The adaptive control is found to offer enhanced robustness to the payload fluctuations (change in mass), which may vary with time, and/or are initially uncertain. For example, adaptive control is applied to the NASA's Helios, a Very Flexible Aircraft (VFA), see Figure 1.1 [21]. These aircrafts are characterized by wings with a large aspect ratio and strong rigid flexible coupling. During flight, the local angle of attack along the wing is expected to change significantly as the wing structure flexes. Furthermore, any uncertainty in the rigid body dynamics

will be exacerbated by uncertainty in the flexible dynamics [21]. Adaptive control is used to deal with the parametric uncertainty. Another huge area of application is in the service industry, where commercial drones are being used in the delivery of packages of varying masses, see Figure 1.2. Amazon is exploring the use of drones for package delivery [22].

Figure 1.1. Adaptive control are used for Very Flexible Aircraft (VFA): NASA's Helios [21].

Figure 1.2. Drones are being used to deliver packages of varying mass [22].

The field of adaptive control is not new. It was conceived during the early 1950s and underwent substantial development during the late 1950s and in the 1960s. In spite of this, the underlying theory and synthesis techniques are not as well known to the average controls engineer as they should be. The reason for this is that the field has been largely associated only with the aerospace industry [23]. For example, as an aircraft flies its mass will slowly decrease as a result of fuel consumption; adaptive control law is used to adapts itself to such changing conditions. Moreover, devices in an aircraft age and wear (e.g., actuator degradation), thus, fault-tolerant flight controllers are needed. Furthermore, there is an uncertain environment around aircrafts such as vortex. An adaptive controller is used to reduce the oscillation caused by the vortex [24]. These problems are amplified in the case of actuator failures, where the aircraft has lost some of its control effectiveness [25].

Recently, a number of industries, including the chemical, pulp and paper, and power industries, have recognized the strength of this field and are now utilizing or trying to utilize it. For example, control of reactor power is important due to safety reasons. In reactor dynamics there are some parameters that vary as a function of the power level, fuel burnup and control rod worth [26]. Therefore, a controller that can adapt to these unknown parameter changes is needed. Also, adaptive control is used for biomass plants, see Figure 1.3 [27]. The changes in humidity and calorific value of fuel, typical of this kind of plant, greatly affects the dynamic response to the power generated, the amount of air used, and the amount of fuel consumed. Additionally, the context of operations changes frequently during the day, principally due to steam blowing for various automatic cleaning processes, filter cleaning and shutdown or reductions in the minimum speed of one of the feed lines for maintenance or excessive biomass humidity. Likewise, the heat transfer loses efficiency with time due to progressive fouling of the boiler [27]. The adaptive

control system is used to attain the production objective of such plants by increasing the stability and reliability of its operation and minimizing the specific consumption of the biomass.

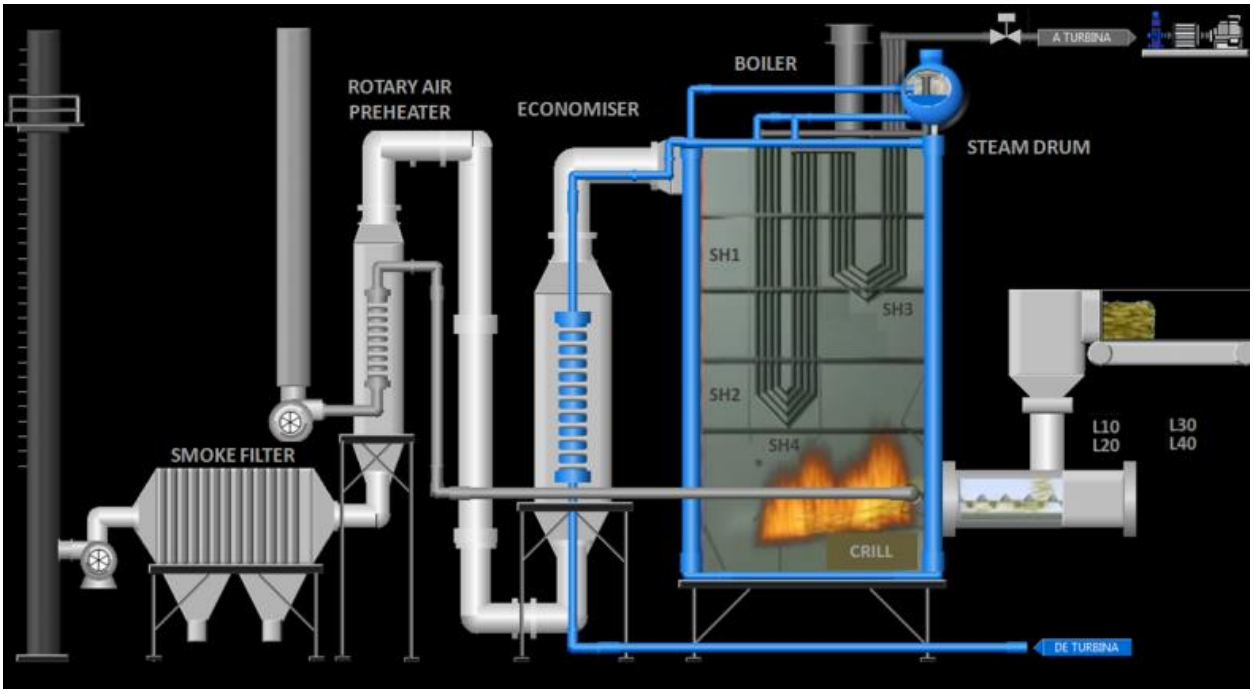


Figure 1.3. Adaptive control system is used for biomass plants: plant at Sangüesa [27].

Figure 1.4. Adaptive control increases the precision of industrial robots [28].

Furthermore, adaptive robot control activates a closed metrology-driven feedback loop that firmly increases the precision of industrial robots [28]. Regardless of whether robots are deployed

for machining, inspection, applying beads or manipulating objects, roboting tasks are consequently executed with 0.1mm absolute positioning accuracy, irrespective of degrading phenomena like play, mechanical flexibility, backlash or thermal effects, see Figure 1.4 [28].

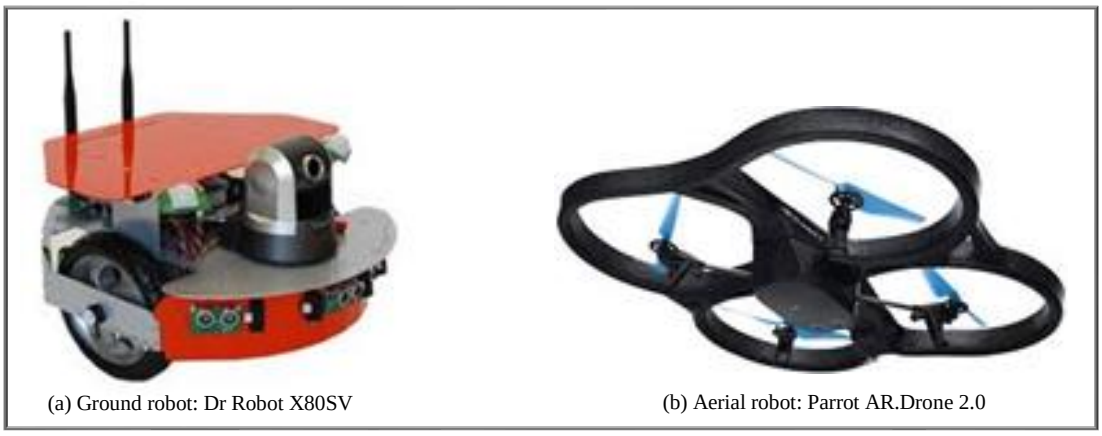
Moreover, the research also demonstrates the design of a robust control for the quadrotor to deal with parameter uncertainty in a linear time-invariant (LTI) time-delay system of the retarded-type, by calculating the real stability radius. This control method is appropriate for processes and systems where the uncertain parameters or disturbances are known or can be determined. The robust control guarantees that if the changes are within given bounds the control law need not to be changed or adapted. Under such circumstance, the robust control approach is preferred since it will be cheaper and less risky to operate compared to the adaptive control. The adaptive closed-loop stability cannot be guaranteed on the same level of confidence as with linear robust controllers under some circumstances. Adaptive controllers often seem to have unsatisfactory transient behavior during adaptation to a plant change (e.g., during a start-up when the adaptive controller is initially wrongly tuned), and they demand highly skilled and educated personnel for tuning and maintenance.

1.3 Systems for Research Work

The systems used in this research are ground and aerial mobile robots. The ground robot, X80SV, shown in Figure 1.5a was manufactured by Dr Robot Inc., and the aerial robot, AR.Drone 2.0, shown in Figure 1.5b was manufactured by Parrot. The two robots come with application program interface (API) reference manuals to aid developers and researchers.

1.3.1 Ground robot: X80SV. The X80SV is a differential drive wheeled mobile robot (DDWMR), with a market value of about $3,000. It is a ready-to-use mobile robot platform designed for researchers developing advanced robot applications such as remote monitoring,

telepresence, and autonomous navigation. The robot comes with a control program written in *C#*, with its graphical user interface (GUI) shown in Figure 1.6. The accompanied control program manually drives the robot to perform translational and rotational motions.



(a) Ground robot: Dr Robot X80SV (b) Aerial robot: Parrot AR.Drone 2.0

Figure 1.5. Mobile robots used in this research work.

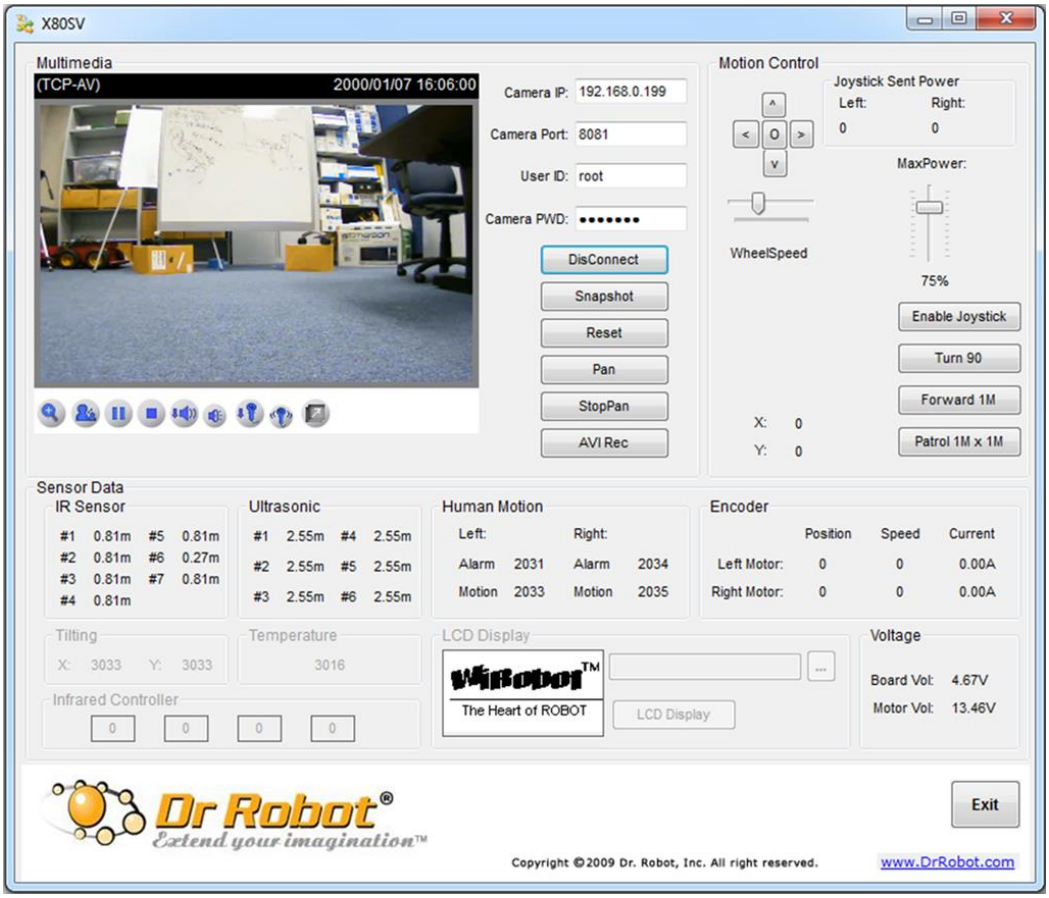


Figure 1.6. Dr Robot X80SV C# program basic interface.

1.3.2 Aerial robot: AR.Drone 2.0. AR.Drone 2.0 is a quadcopter (referred to as quadrotor) type of unmanned aerial vehicle (UAV) built by the French company Parrot. It has a market value of about $300 and is commercially designed for playing games. It is accompanied by a program written in C, and it is designed to be controlled by mobile or tablet operating systems such as iOS or Android. Figure 1.7 shows an iPhone device being used to control the drone. Individuals and institutions realized because it is cheap and its low cost of maintainability, it will be an economical and reliable device for research.

Figure 1.7. iPhone device being used to control Parrot AR.Drone 2.0 [29].

1.4 Major Reported Work

1.4.1 Ground robot: X80SV. The different parts of this section of the research work have been published in peer-reviewed journals and conference papers, and some are discussed here. Firstly, DDWMR autonomous navigation has been presented [30], where fuzzy logic has been used to build behavior-based fuzzy controller for the navigation algorithm. It is based on

behavioral architecture which can deal with uncertainties in unknown environments and has the ability to accommodate different behaviors. The algorithm drives a robot to reach a target while avoiding obstacles in the environment. The proposed control algorithm was simulated and experimented using Voyager II robot in a laboratory environment (see Figure 1.8), with a start point of $(0m, 0m)$ and goal point of $(3m, 5m)$. The robot reaches the goal in almost 20 seconds.

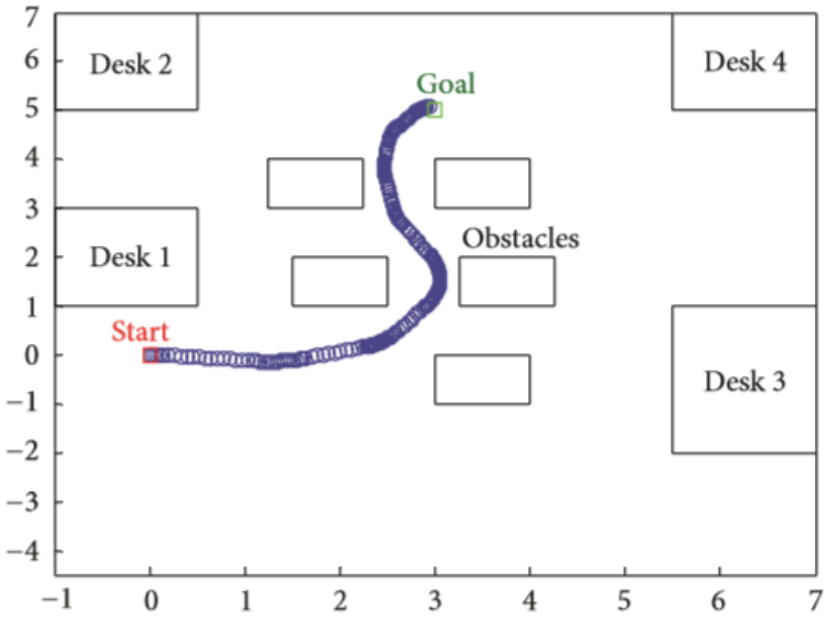


Figure 1.8. Voyager II DDWMR avoids obstacles and reaches the goal [30].

Another approach presented uses model predictive equilibrium point control (MPEPC) [31], where the control algorithms were developed based on the discretized model of the system, on a carefully designed compact parameterization of a rich set of closed-loop trajectories, and the use of expected values in cost definitions. The framework combined the ideas of time-optimality and equilibrium point control to generate quick and responsive solutions. The proposed algorithm was tested in an indoor environment, a tight L-shaped corridor, using pedestrians as dynamic obstacles.

1.4.2 Aerial robot: AR.Drone 2.0. The different parts of this section of the research work have also been published in peer-reviewed journals and conference papers, and some are discussed

here. Firstly, modeling and altitude control of a quadrotor UAV has been presented [32], where the quadrotor model was derived using the Euler-Lagrange equations, and experiment performed to identify the model parameters. The altitude controller was designed using the dynamic surface control (DSC) method, with total thrust as the control input to the plant. The proposed control law was derived based on the nonlinear altitude dynamics and simulations conducted.

Secondly, stabilized PID controllers have also been designed for controlling quadrotor's attitude and altitude under disturbance and noisy conditions [33]. A linearized altitude model was obtained by combining the first-order transfer function of motor dynamics and the nonlinear altitude equation of motion. The altitude model derivation approach used was not convincing. The closed-loop system behaviors were analyzed while the quadrotor is hovering in the presence of an unknown disturbance by using Extended Kalman Filter (EKF) to address it.

Thirdly, quadrotor control has also recently been presented [34], where feedback linearization and model reference adaptive control (MRAC) are integrated to design attitude control system for a fixed wing UAV. In order to demonstrate the performance of attitude control system, adaptive and PID control laws are used for the coupling nonlinear simulation model.

Fourthly, direct and indirect MIT rule MRAC has been applied to a lightweight low-cost quadrotor UAV platform [25]. A baseline trajectory controller was augmented by the adaptive controller. The research based the adaptive controller on Lyapunov stability. The adaptive controller was found to offer increased robustness to parametric uncertainties over the fixed-gain approach. Particularly, it was found to be effective in mitigating the effects of a loss-of-thrust anomaly, which may occur due to component failure or physical damage, e.g., actuator failure. The controllers were simulated and flight testing carried out in an indoor test facility using baseline, direct MRAC, and indirect (combined/composite) MRAC, see Figure 1.9 for altitude

responses. Loss-of-thrust uncertainty occurs at approximately $26s$, denoted by asterisk and the arrow [25]. Flight tests were conducted in the RAVEN testbed at MIT. The adaptive controller was implemented in C^{++}, and the commands were sent to the quadrotor using a USB wireless remote-control module.

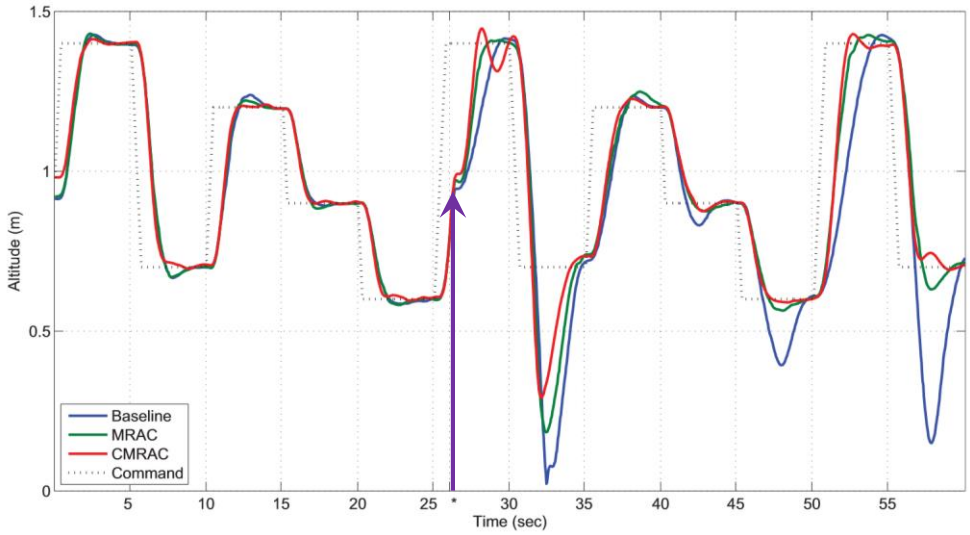


Figure 1.9. Altitude tracking of the baseline, MRAC, and CMRAC controllers [25].

Furthermore, the effects of time delays on quadrotor-type helicopter dynamics has been studied [35]. The research in the article used numerical and simulation to show that for large delays ($> 0.20s$), quadrotor system's response might not be stabilized or converged due to increased torque and position drift increases, and this poses a significant challenge.

Moreover, the problem of designing a robust state feedback control that provides global stability for an uncertain nonlinear system has been the subject of considerable research over the last decade. Robust control of nonlinear systems with parametric uncertainty has been studied [36], where probabilistic robustness analysis and synthesis for nonlinear systems with uncertain parameters are presented. Monte Carlo simulation was used to estimate the likelihood of system instability and violation of performance requirements subject to variations of the probabilistic system parameters.

Lastly, the stability of LTI time-delay systems of the retarded and neutral types subjected to real perturbations have been investigated [37]. The paper presents a readily computable formulae for the real stability radius with respect to an arbitrary stability region in the complex plane for both linear continuous and discrete time-delay systems.

1.5 Objectives of Research Work

The main goal of this research is to contribute to the field of robotics and control theory by developing adaptive and robust controllers for stabilized and reliable autonomous navigation of mobile robots. Specific objectives include:

- Derive dynamic models for X80SV (ground) and AR.Drone 2.0 (aerial) motions
- Develop control systems to improve stability to satisfy a desired criteria
- Study and calculate nonlinear effects of saturation, time delay, and system uncertainty
- Experiment to demonstrate the robustness of the control systems
- Validate control design through analytical, numerical simulation, and experiment using MATLAB/Simulink

1.6 Major Contributions

The major contributions of this research work include:

- Developed an effective autonomous navigation control algorithm for a DDWMR
- Identified and built quadrotor's altitude models for control
- Implemented quadrotor nonlinear kinematics and dynamics using MATLAB S-function
- Studied and estimated the time delay in a quadrotor control system
- Studied the stability of a parametric perturbed retarded-type delay system by calculating the stability radius

- Developed adaptive and robust altitude autonomous controllers for a quadrotor considering saturation, time delay, and parametric uncertainty
- Developed an interactive MATLAB GUI and Simulink Programs for the control of X80SV and AR.Drone 2.0

1.7 Problem Solution and Scope

1.7.1 Ground robot: X80SV. The solution involved designing and implementing PID feedback controllers for individual robot behavior, and then developing control navigation system based on the individual behaviors. The approach used for the navigation system made use of a low-level strategy to build a model based on the continuous kinematics of the DDWMR, and then using logic control, hybrid automata approach, a behavior-based navigation architecture is developed. The controllers are designed based on the nonlinear kinematics of the DDWMR. MATLAB GUI is developed as an interface for the implementation.

1.7.2 Aerial robot: AR.Drone 2.0. Firstly, the solution involved designing and implementing autonomous position (x, y, and z) and attitude (roll, pitch, and yaw) control for quadrotor. The above is achieved based on the nonlinear dynamics and kinematics of the quadrotor. MATLAB-based simulator using PD-feedback control is developed to achieve stabilization.

Secondly, after the above has been achieved, altitude models are obtained using the following two approaches: (1) the model derived using white-box approach based on the dynamics and kinematics of the quadrotor and (2) the model built using black-box approach based on data collected from experiments using the drone. Controllers such as full-state feedback (pole placement or proportional plus velocity (PV), and linear quadratic regulator (LQR)), and MRAC are designed and validated through simulations and experiments using MATLAB/Simulink. Then,

in order to obtain the adaptive and robust controllers, the nonlinear effects of control input saturation, time delay in the controlled system, and system parametric uncertainty are studied and addressed. MATLAB GUI and Simulink programs are developed for the implementation.

CHAPTER 2

Background and Literature Review

This chapter provides a brief background of mobile robots, their classifications and types of behaviors and navigations. This is followed by a description of the various modeling and control theories used for the research. It then continues with reviewing the types of system responses and its transient properties. The chapter ends with a discussion on nonlinear effects of saturation, time delay, and system uncertainty inherent in aerial robots.

2.1 Mobile Robots and their Classifications

Mobile robots have the capability to move around in their environment, they are not supposed to be fixed to one physical location [15]. By contrast, industrial robots are usually more-or-less stationary, consisting of a jointed arm (multi-linked manipulator) and gripper assembly (or end effector) attached to a fixed surface [15]. Mobile robots come in many shapes and sizes, but they all contain three main components. First, there is some combination of sensors that are used for understanding the environment. Then, there is the onboard computer for planning and decision making. Finally, a form of locomotion allowing the robot to act in its environment [18].

The centerpiece of every robotic control system is an onboard controller. It makes decisions based on the available sensor data and sends instructions to its motors to control the robot's movement. Robotic control systems require the following subsystems [18]: (1) an interface to input/output (I/O) - robotic control systems have to communicate with a wide variety of sensors and actuators. Key sensors such as LIDAR and GPS commonly use a USB or serial interface, while motors might require a digital port or CAN interface, (2) low-level control - PID loops or state-space equations are implemented to perform processing based on sensor feedback. For example, a PID loop is used to process the encoder feedback and navigate the robot in a straight

line. This type of control requires deterministic response and tight integration with I/O, (3) autonomous navigation system - a mobile robot has subsystems for perception and planning. Once a robot perceives, or understands its sensor data, the data is passed to a higher-level planning module. The planning module can be broken down even further; low-level planning, such as stopping when an obstacle is present, or high-level planning, such as making decisions regarding the mission of the robot, and (4) user interface - often used to display information regarding a robot's health, such as power consumption levels, and a notification of hardware failures. This includes both remote and local APIs.

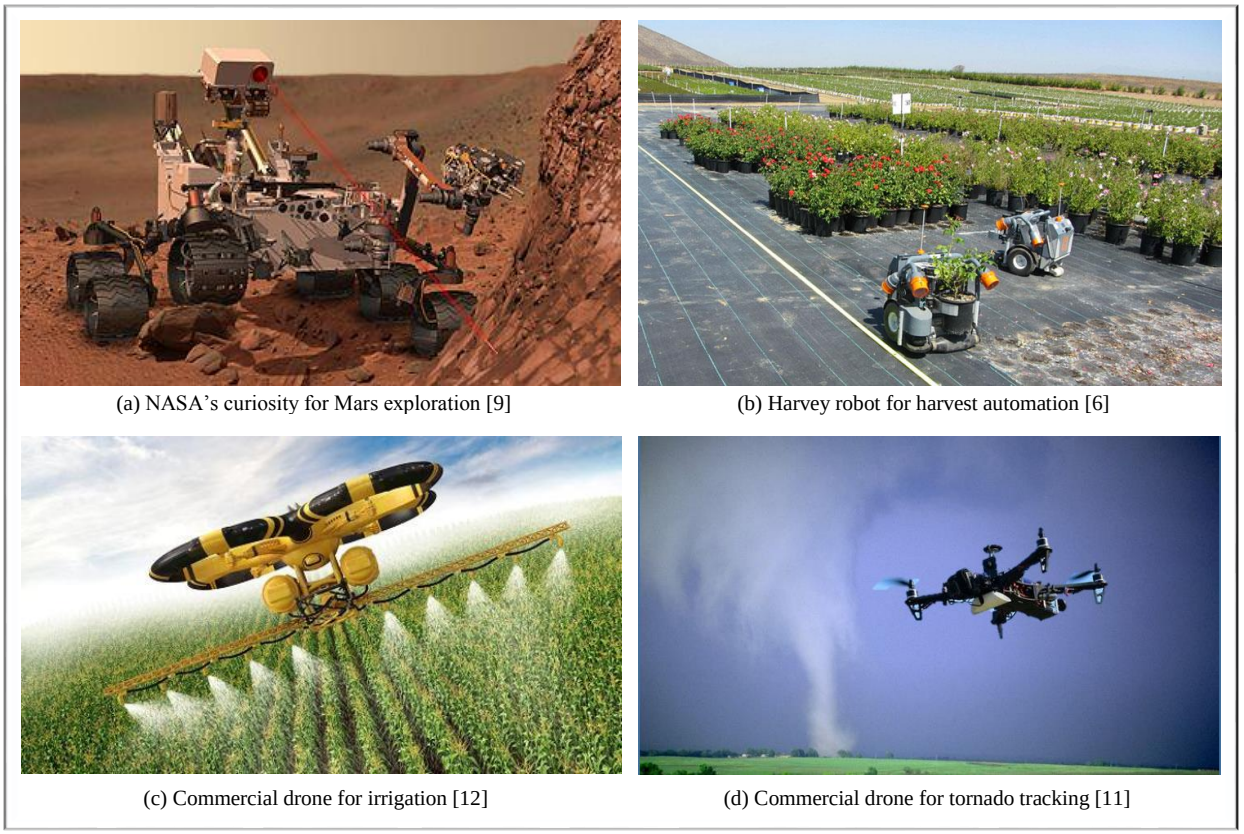
Mobile robots are found in places such as industrial, military, and security settings. Domestic robots are consumer products, including entertainment robots and those that perform certain household tasks such as vacuuming or gardening [15]. Mobile robots may be classified by (see Figure 2.1 for examples) [15]:

- The environment in which they travel, include:
    - Land or home robots usually referred to as unmanned ground vehicles (UGVs): They are most commonly wheeled or tracked, but also include legged robots with two or more legs (humanoid, or resembling animals or insects),
    - Aerial robots are usually referred to as unmanned aerial vehicles (UAVs),
    - Underwater robots are usually called autonomous underwater vehicles (AUVs),
    - Polar robots, designed to navigate icy, crevasse filled environments.
- The device they use to move, mainly:
    - Legged robot: human-like legs (i.e. an android) or animal-like legs,
    - Wheeled robot,
    - Tracks.

(a) Car-like [5] (b) Snake [13] (c) Cheetah [14]

(d) Personal transportation [3] (e) Quadrotor [4] (f) Humanoid [1]

Figure 2.1. Types of mobile robots.



(a) NASA's curiosity for Mars exploration [9] (b) Harvey robot for harvest automation [6]

(c) Commercial drone for irrigation [12] (d) Commercial drone for tornado tracking [11]

Figure 2.2. Applications of mobile robots.

Mobile robots are most commonly found performing tasks that are fun, dull, dirty, or dangerous. The military uses robotic systems for dangerous tasks, such as walking through minefields, deactivating bombs, or clearing out hostile buildings [18]. Farmers use mobile robots to perform dirty tasks, including harvesting, collecting crop data, weeding, and micro spraying [18]. Hospitals use mobile robots to deliver specimens to laboratories and for assistive care [18]. Mobile robots are even used to perform routine chores around the house, such as vacuuming and cleaning pools, gutters, and lawn mowing [18]. Despite challenges, there are several other wide applications of mobile robots that include security and surveillance, planetary exploration, search and rescue mission, and inspection, see Figure 2.2 for examples.

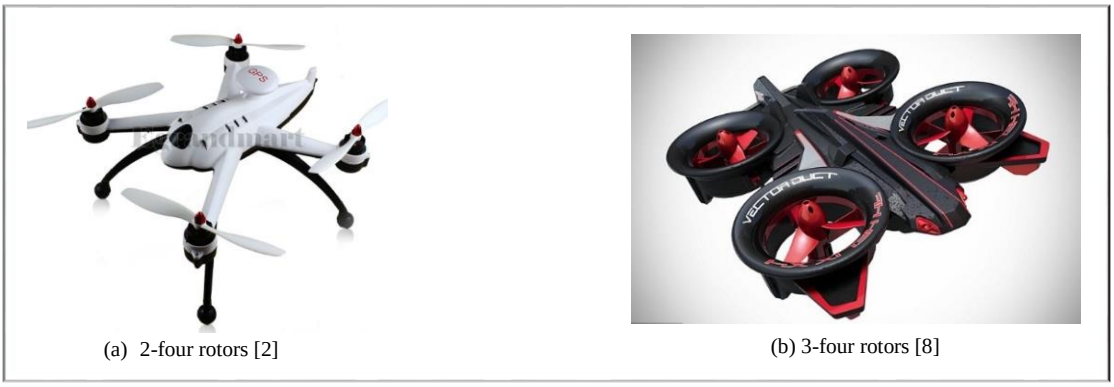
2.1.1 Ground robot: X80SV. Wheeled mobile robots (WMRs) are increasingly present in industrial and service robotics, particularly when flexible motion capabilities are required on reasonably smooth grounds and surfaces. Several mobility configurations (wheel number and type, their location and actuation and single- or multi-body vehicle structure) can be found in different applications [38]. The most common for single-body robots are differential drive and synchro drive (both kinematically equivalent to a unicycle), tricycle or car-like drive and omnidirectional steering [38]. Figure 2.3 shows some WMR configurations.

(a) Six wheeled platform [10] (b) Two wheeled platform [7]

Figure 2.3. Types of wheeled mobile robots.

2.1.2 Aerial robot: AR.Drone 2.0. During the last decade, with the advancement in relevant technology, the demand for flying mobile robots or UAVs has rapidly increased. UAVs

emergence also has to do with the simplicity of their construction and maintenance, their ability to hover, and their vertical take-off and landing (VTOL) capability.



(a) 2-four rotors [2]

(b) 3-four rotors [8]

Figure 2.4. Types of unmanned aerial vehicles (UAVs).

Rotorcraft UAVs have a variety of configurations that include a conventional helicopter with a main and tail rotor, a coax with counter-rotating coaxial rotors, and quadrotors [39], see Figure 2.4 for examples. Quadrotor types of UAVs have four rotors that are controlled independently. The movement of the quadrotor results from changes in the speed of each rotor.

2.2 Types of Mobile Robot Behaviors and Navigations

Control systems are supposed to have navigation control algorithms that will make mobile robots successfully accomplish different motions and behaviors, which include translational (x, y, and z), rotational (roll, pitch, and yaw), go-to-goal, avoid-obstacles (stationary or moving), follow a path (e.g. line, circle), follow-wall, track-target, using combined skills (e.g. navigation system), image and audio recognition, etc.

For any mobile device, the ability to navigate in its environment is important. Avoiding dangerous situations such as collisions and unsafe conditions (temperature, radiation, exposure to weather, etc.) comes first, but if the robot has a purpose that relates to specific places in the robot environment, it must find those places [40]. Navigation can be defined as the combination of the

three fundamental competences: (1) Self-localization, (2) Path planning, and (3) Map-building and map interpretation [40]. There are many types of mobile robot navigations, which include manual remote or tele-operated, guarded tele-operated, line-following car, autonomously randomized robot, autonomously guided robot, and sliding autonomy [41].

2.3 Modeling and Control Theories

Fundamental to the successful, autonomous operation of mobile robots are robust motion control algorithms. Control algorithms determine the appropriate action to take (actuators) based on the current state (sensors). Developing the control algorithms poses a significant challenge that includes noisy sensors, nonlinearity, dynamic environment, non-holonomic drive, etc. This section provides a brief background and description for various control design theories and modeling techniques used in this research work.

In control engineering, a state-space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations [42]. If the dynamical system is LTI, the differential and algebraic equations may be written in matrix form, and the state-space representation (also known as the "time-domain approach") provides a convenient and compact way to model and analyze systems with multiple inputs and outputs. Consider the basic form of the state-space representation of a continuous LTI system given as [42]

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{2.1}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and $\mathbf{D}$ are the state-space parameters: system (state) matrix, input matrix, output matrix, and feedforward (feedthrough) matrix respectively. $\mathbf{x}(t)$ is the state vector, $\mathbf{u}(t)$ is the control (input) vector, and $\mathbf{y}(t)$ is the output vector. The size of $\mathbf{A}$ is n x n, $\mathbf{x}(t)$ is n x 1, $\mathbf{B}$ is n x m, $\mathbf{u}(t)$ is m x 1, $\mathbf{y}(t)$ is q x 1, $\mathbf{C}$ is q x n, and $\mathbf{D}$ is q x m.

2.3.1 Controllability and observability. A system is said to be controllable at time t_o if it is possible to transfer the system from any initial state $\mathbf{x}(t_o)$ to any other state in a finite interval of time by means of an unconstrained control vector [43]. A system is said to be observable at time t_o if it is possible to determine the state $\mathbf{x}(t_o)$ of the system from the observation of the output over a finite time interval [43].

The concepts of controllability and observability were introduced by Kalman [42]. They play an important role in the design of control systems in state space. In fact, the conditions of controllability and observability may govern the existence of a complete solution to the control design problem. Although most physical systems are controllable and observable, corresponding mathematical models may not possess the properties of controllability and observability [43]. It is then necessary to know the conditions under which the system is controllable and observable. This section primarily deals with controllability and observability.

2.3.1.1 Controllability. The LTI system in equation (2.1) is said to be completely state controllable if and only if the rank of the controllability matrix, $CONT$, given in equation (2.2), is n, which is the dimension of the system matrix, **A** [43].

$$CONT = [\mathbf{B} \vdots \mathbf{AB} \vdots \cdots \vdots \mathbf{A}^{n-1}\mathbf{B}] \tag{2.2}$$

If the system is controllable then it implies it is stabilizable. For a partially controllable system, if the uncontrollable modes are stable and unstable modes are controllable, then the system is still said to be stabilizable [43].

2.3.1.2 Observability. The LTI system in equation (2.1) is said to be completely observable if and only if the rank of the observability matrix, $OBSER$, given in equation (2.3), is n, which is the dimension of the system matrix, **A** [43].

$$OBSER = [\mathbf{C} \vdots \mathbf{CA} \vdots \cdots \vdots \mathbf{CA}^{n-1}]' \tag{2.3}$$

If the system is observable then it implies it is detectable. For a partially observable system, if the unobservable modes are stable and observable modes are unstable, then the system is still said to be detectable [43].

2.3.2 Proportional-integral-derivative (PID). The traditional PID-feedback controller is given by [42]

$$\mathbf{u}(t) = \mathrm{PID}(\mathbf{e}) = K_P\mathbf{e}(t) + K_I\int_0^t \mathbf{e}(\tau)d\tau + K_D\frac{d\mathbf{e}(t)}{dt} \tag{2.4}$$

where $\mathbf{e}$ is the error between the desired value and the output value, K_P is the proportional gain, K_I is the integral gain, K_D is the derivative gain, and t is time. The tuning parameters K_P, K_I, and K_D are determined by using classical control strategy such as root locus based on a design specification, or they can just be tuned to obtain a suitable plant response.

The controller attempts to minimize the error by adjusting the system through use of a manipulated variable. The P controller depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change [44]. The weighted sum of these three actions is used to adjust the system via a control element such as the position of a control valve, the position of a mobile robot, or the power supplied to a heating element.

In general, increasing K_P independently will increase the maximum overshoot, decrease the rise time and the steady-state error, degrade the stability, with a small change in the settling time. Increasing K_I independently will increase the maximum overshoot, decrease the rise time, eliminate the steady-state error, degrade the stability, and increase the settling time. Increasing K_D independently will decrease the maximum overshoot, cause minor change in the rise time, no effect in theory on the steady-state error, will improve stability if K_D is small, and decrease the settling time [44].

2.3.3 Full-state feedback (FSF). This is a method employed in feedback control system theory to place the closed-loop poles of a plant in pre-determined locations in the s-complex plane, and it is assumed that all of the state variables are measurable and are available for feedback, that the system is completely state controllable, and the plant has to be linear [43]. The system is stabilized or regularized by selecting the control signal, $\mathbf{u}(t)$, as [43]

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) + \mathbf{r}(t) \tag{2.5}$$

where $\mathbf{r}(t)$ is the reference input and $\mathbf{K} = [k_1 \ k_2 \ ... \ k_n]$ is a matrix of control gains. Substituting (2.5) into (2.1), and rearranging, the modified, closed-loop, system is given as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}(t) + \mathbf{B}\mathbf{r}(t) \\ \boldsymbol{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{2.6}$$

This system is stable if $\mathbf{K}$ can be determined such that the roots of characteristic equation, $|s\mathbf{I} - \mathbf{A} + \mathbf{B}\mathbf{K}| = 0$, lies in the left hand side of the s-complex plane. There are two main methods of the full-state feedback for determining the control gains, pole placement and LQR. The schematic of the FSF control system design to be used is shown Figure 2.5.

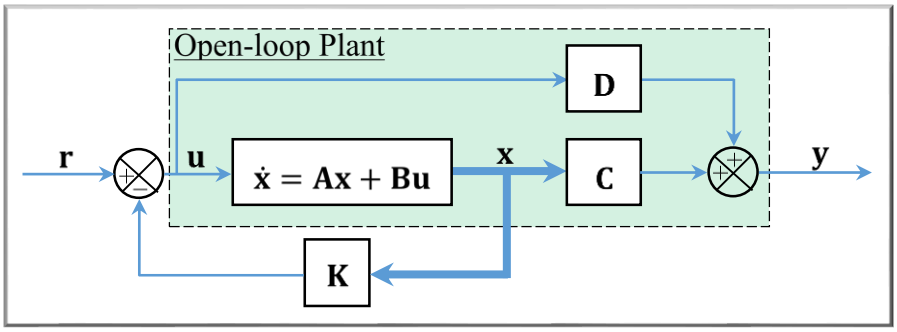


Figure 2.5. Schematic of the full-state feedback control system [43].

2.3.3.1 Pole placement. This method is based on the fact that if the system is completely state controllable, then poles of the closed-loop system may be placed arbitrarily at any desired locations in the s-complex plane. For example, for a second-order plant ($n = 2$), if the desired

closed-loop poles, $I = \beta_1 \pm \beta_2 j$, are placed in the left-half of the s-complex plane, then the closed-loop control system will be asymptotically stable for all $\mathbf{x}(t) \neq \mathbf{0}$ [43]. The characteristic polynomial of the closed-loop system, from equation (2.6), $|s\mathbf{I} - \mathbf{A} + \mathbf{BK}|$, is equated to the desired characteristic polynomial as

$$|s\mathbf{I} - \mathbf{A} + \mathbf{BK}| = \big(s - (\beta_1 + \beta_2 j)\big)\big(s - (\beta_1 - \beta_2 j)\big) \tag{2.7}$$

Thus, by equating the coefficients of like powers of s on both sides, $\mathbf{K} = [k_1 \ k_2]$ can be determined. The pole placement controller is also termed PV when it is being applied on the system's transfer function. PV control is different from PD control, in the sense that it does not yield numerator dynamics.

2.3.3.2 Linear quadratic regulator (LQR). The theory of optimal control is concerned with operating a dynamic system at minimum cost. The case where the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic function is called the *LQ problem*. The LQR is an important part of the solution to the *linear quadratic Gaussian (LQG)* problem. LQR has an advantage over the pole placement method, in that it provides a systematic way of computing the state feedback gain matrix, and also the designed closed-loop systems is always stable [43].

In effect, the LQR algorithm takes care of the tedious work done by the control systems engineer in optimizing the controller. However, the engineer still needs to specify the weighting factors and compare the results with the specified design goals. Often this means that controller synthesis will still be an iterative process where the engineer judges the produced optimal controllers through simulation and then adjusts the weighting factors to get a controller more in line with the specified design goals. Difficulty in finding the right weighting factors limits the application of the LQR based controller synthesis. Given the system in (2.6), LQR determines the

optimal controller gain matrix, **K**, of the optimal control input $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$ so as to minimize the performance index, a quadratic cost function, given as [43]

$$J = \int_0^\infty [\mathbf{x}'(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}'(t)\mathbf{R}\mathbf{u}(t) + 2\mathbf{x}'(t)\mathbf{N}\mathbf{u}(t)]\, dt \tag{2.8}$$

where the Hermitian matrices, $\mathbf{N} \geq \mathbf{0}, \mathbf{Q} \geq \mathbf{0}$, and $\mathbf{R} > \mathbf{0}$ to be selected, are weighting parameters, which define the trade-off between error (how fast the output $\mathbf{y}(t)$ tracks the reference command $\mathbf{r}(t)$) and the expenditure of the control effort or energy. **K** is determined by [43]

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P} \tag{2.9}$$

where the positive-definite matrix, **P**, in (2.9) must satisfy the reduced-matrix Riccati equation given as [43]

$$\mathbf{A}^{\mathrm{T}}\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{2.10}$$

where **N = 0**. Thus, by solving equation (2.10) for **P**, **K** can be determined, and therefore $\mathbf{u}(t)$ can be obtained.

2.3.4 Adaptive control (AC). The unknown and unmeasurable variations of a process or system parameters can degrade the performance of the control systems. Similarly to the disturbances acting upon the controlled variables, one can consider that the variations of the process parameters are caused by disturbances acting upon the parameters (called parameter disturbances), which also affect the performance of the control systems [45]. AC is the control method used by a controller which must adapt to a controlled system with parameters which vary, or are initially uncertain [46].

AC is different from *robust control (RC)* in that it does not need a priori information about the bounds on these uncertain or time-varying parameters. RC guarantees that if the changes are within given bounds the control law need not be changed, while AC is concerned with control law

changing with time. [46]. The foundation of AC is parameter estimation. Common methods of estimation include recursive least squares and gradient descent. Both of these methods provide update laws which are used to modify estimates in real time (i.e., as the system operates). Lyapunov stability is used to derive these update laws and show convergence criterion (typically persistent excitation). Projection (mathematics) and normalization are commonly used to improve the robustness of estimation algorithms. AC is also called *adjustable control* [46].

An AC system measures a certain performance index of the control system using the inputs, the states, the outputs, and the known disturbances. From the comparison of the measured performance index and a set of given ones, the adaptation mechanism modifies the parameters of the adjustable controller and/or generates an auxiliary control. This is done in order to maintain the performance index of the control system close to the set of given ones (i.e., within the set of acceptable ones) [45].

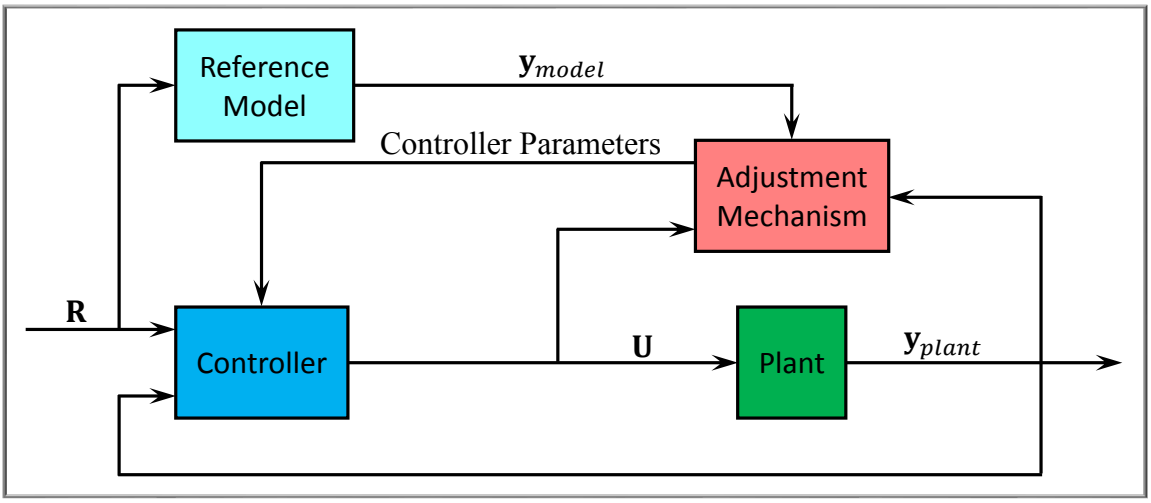


Figure 2.6. Typical MRAC setup [47].

An AC system, which contains in addition to a feedback control with adjustable parameters a supplementary loop acting upon the adjustable parameters of the controller, will monitor the performance of the system in the presence of parameter disturbances [45]. In general, AC can be

classified as feedforward adaptive control or feedback adaptive control and direct method or indirect method [46]. Direct methods are ones wherein the estimated parameters are those directly used in the adaptive controller. In contrast, indirect methods are those in which the estimated parameters are used to calculate required controller parameters [46]. There are several broad categories of feedback adaptive control, and in this research work the model reference adaptive control (MRAC) will be applied. There are different types of MRAC depending on the adjustable mechanism, and one of the modern ones is the MIT rule. Typical setup of MRAC is shown in Figure 2.6.

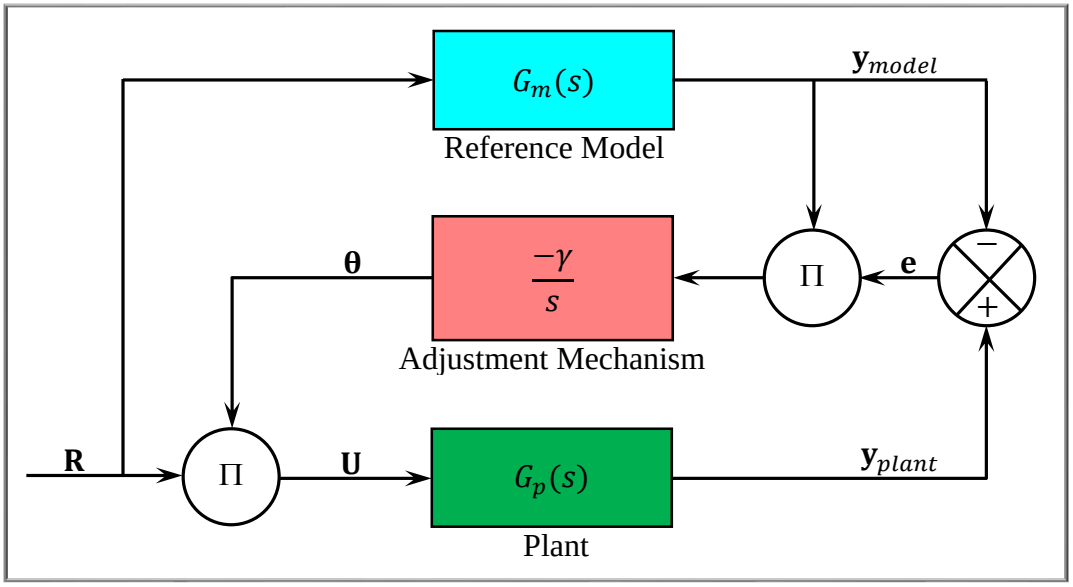


Figure 2.7. MIT rule MRAC closed-loop system [47].

2.3.4.1 MIT rule MRAC. Consider the MIT rule MRAC setup shown in Figure 2.7. The tracking error, **e**, is given by [47]

$$\mathbf{e} = \mathbf{y}_{plant} - \mathbf{y}_{model} = \ G_p\mathbf{U} - \ G_m\mathbf{R} \tag{2.11}$$

Next is to define a cost function, J, in terms of **e**, and one and simplest way is to expressed its as [47]

$$J(\boldsymbol{\theta}) = \frac{1}{2}\mathbf{e}^2(\boldsymbol{\theta}) \tag{2.12}$$

where $\boldsymbol{\theta}$ is the updating parameter. The rule employs a feed-forward gain adaption to update $\boldsymbol{\theta}$, as shown below [47]

$$\frac{d\boldsymbol{\theta}}{dt} = -\gamma \frac{\partial J}{\partial \boldsymbol{\theta}} = -\gamma \mathbf{e} \frac{\partial \mathbf{e}}{\partial \boldsymbol{\theta}} \tag{2.13}$$

where γ is a tuning parameter and $\partial \mathbf{e}/\partial \boldsymbol{\theta}$ is sensitivity derivative. It is important to note that the MIT rule by itself does not guarantee error convergence or stability. An MRAC designed using the MIT rule is very sensitive to the amplitudes of the signals. As a general rule, the value of γ is kept small. Tuning of γ is crucial to the adaptation rate and stability of the controller [47].

2.3.5 Robust control (RC). This is a branch of control theory which approach to controller design explicitly deals with uncertainty [48]. RC methods are designed to function properly provided that uncertain parameters or disturbances are found within some (typically compact) set. Robust methods aim to achieve robust performance and/or stability in the presence of bounded modeling errors. In contrast with an AC policy, RC policy is static; rather than adapting to measurements of variations, the RC controller is designed to work assuming that the uncertainties are known [48].

The theory of RC began in the late 1970s and early 1980s and soon developed a number of techniques for dealing with bounded system uncertainty [48]. Probably the most important example of a RC technique is *H-infinity loop-shaping*; this method minimizes the sensitivity of a system over its frequency spectrum, and this guarantees that the system will not greatly deviate from expected trajectories when disturbances enter the system. An emerging area of RC from application point of view is *sliding mode control (SMC)* which is a variation of *variable structure control (VSC)* [48]. Robustness property of SMC towards matched uncertainty as well as the simplicity in design attracted a variety of application. Another example is *loop transfer recovery (LQG/LTR)*, which was developed to overcome the robustness problems of LQG control. Other

robust techniques includes *quantitative feedback theory (QFT), gain scheduling, back stepping, feedback linearization,* etc. [48].

2.3.6 Root locus design and SISO design tool. A common technique for meeting control design criteria is root locus design. This approach involves iterating on a design by manipulating the compensator gain, poles, and zeros in the root locus diagram [49]. The technique consists of plotting the closed-loop pole trajectories of a feedback system in the complex plane as system parameter, k, varies over a continuous range of values varies. The plot can be used to identify the gain value associated with a desired set of closed-loop poles [49].

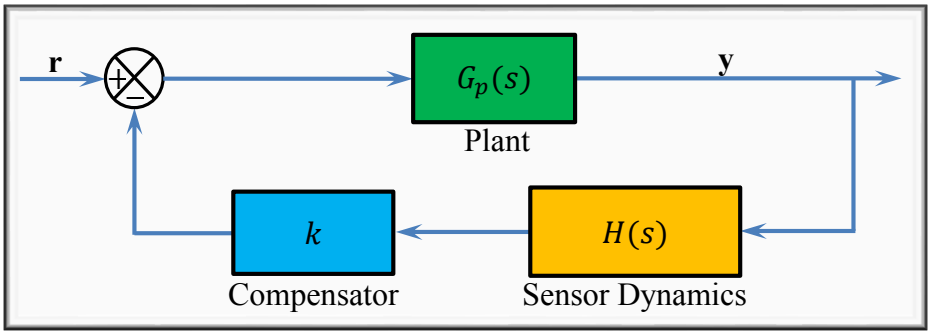


Figure 2.8. Root locus used to design the tracking loop system parameter, k.

Typically, the root locus method is used to tune the loop gain of a single-input-single-output (SISO) control system by specifying a designed set of closed-loop pole locations. Consider, for example, the tracking loop shown in Figure 2.8, where $G_p(s)$ is the plant, $H(s)$ is the sensor dynamics, and k is a scalar gain to be adjusted. The closed-loop poles are the roots of [49]

$$1 + kG_p(s)H(s) = 0 \tag{2.14}$$

The root locus technique, available in MATLAB, consists of plotting the closed-loop pole trajectories in the complex plane as k varies. The plot can be used to identify the gain value associated with a desired set of closed-loop poles [49].

MATLAB has a SISO design tool that allows the design of a SISO compensator using root locus, Bode diagram, Nichols, and Nyquist techniques (e.g., see Figure 2.9) [50]. The tool is an interactive GUI for automatic designing of a compensator. The compensating tuning can be done directly using the SISO design task node (see Figure 2.9) or using the control and estimation tools manager and the graphical tuning window (see Figure 2.10). The MATLAB command to access this tool is *sisotool(.)* [50].

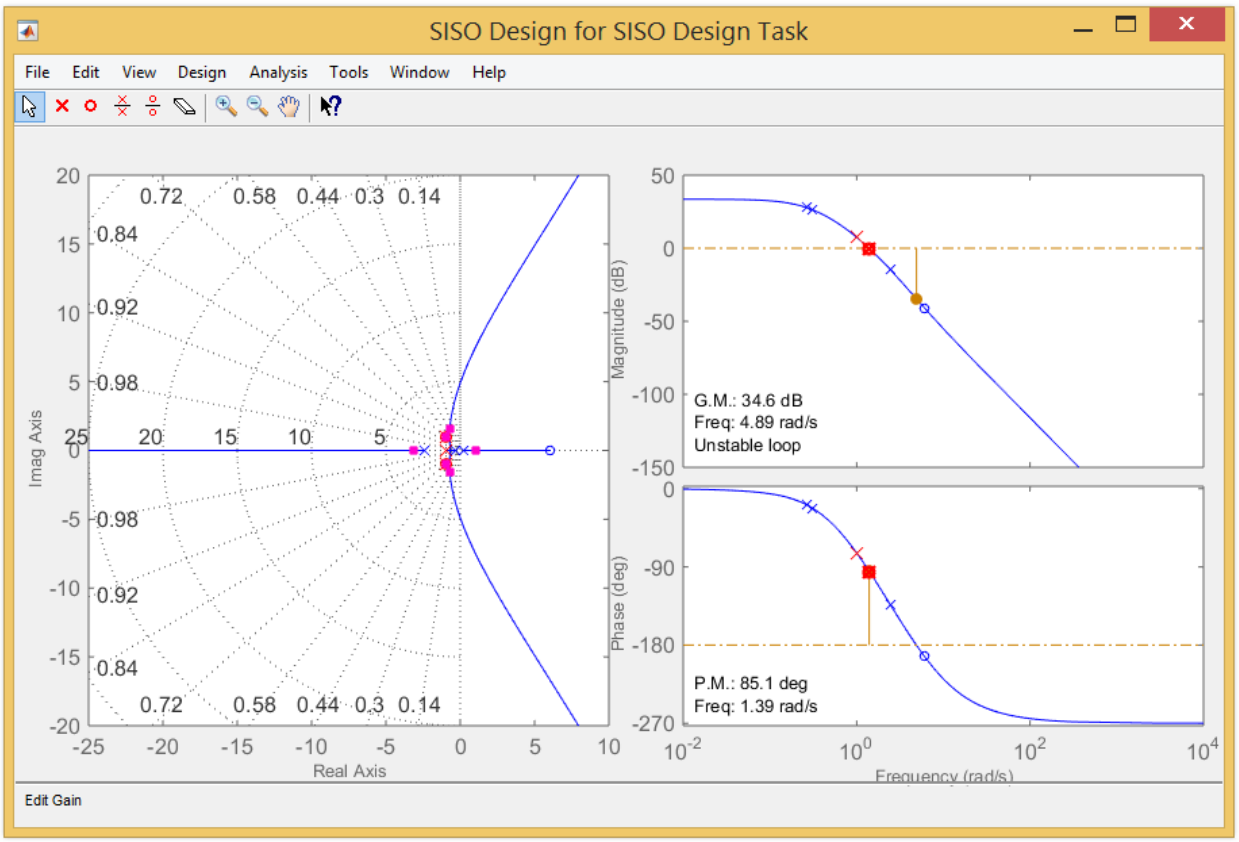


Figure 2.9. An example of root locus plot using SISO design tool for 3rd order system [50].

2.3.7 System modeling and identification. System identification is the art and science of building mathematical models of dynamic systems from observed input-output, measured, data [51]. It can be seen as the interface between the real world of applications and the mathematical world of control theory and model abstractions. A dynamical mathematical model in this context

is a mathematical description of the dynamic behavior of a system or process in either the time or frequency domain. Examples include: physical processes such as the movement of a falling body under the influence of gravity, or economic processes such as stock markets that react to external influences [51].

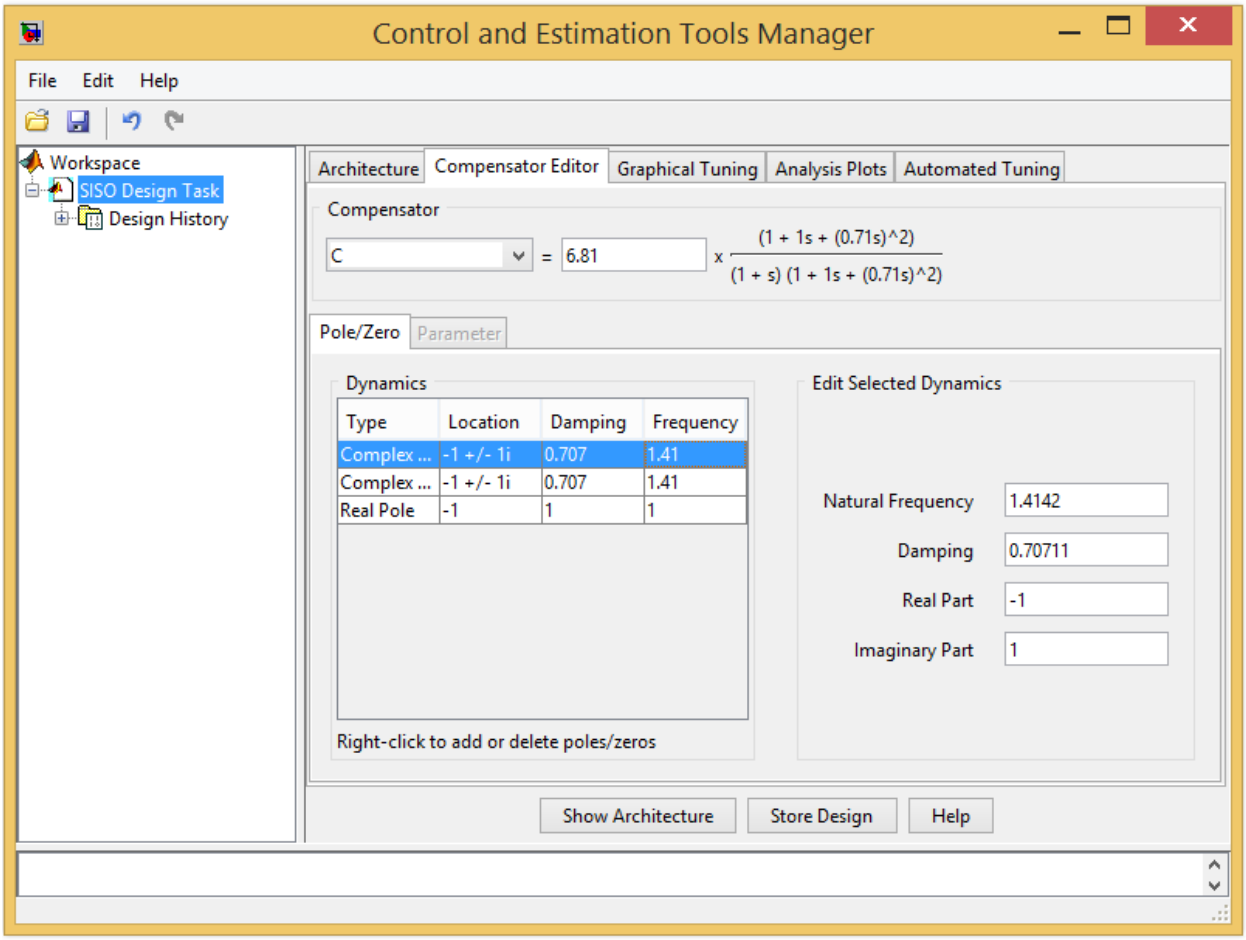


Figure 2.10. SISO design control and estimation tools manager [50].

One could build a so-called *white-box model* based on first principles, e.g. a model for a physical process from the Newton equations, but in many cases such models will be overly complex and possibly even impossible to obtain in reasonable time due to the complex nature of many systems and processes [51]. A much more common approach is therefore to start from measurements of the behavior of the system and the external influences (inputs to the system) and

try to determine a mathematical relation between them without going into the details of what is actually happening inside the system. Two types of models are common in the field of system identification [51]:

- *Grey-box model*: although the peculiarities of what is going on inside the system are not entirely known, a certain model based on both insight into the system and experimental data is constructed. This model does however still have a number of unknown free parameters which can be estimated using system identification. Grey-box modeling is also known as *semi-physical modeling*.
- *Black-box model*: No prior model is available. Most system identification algorithms are of this type.

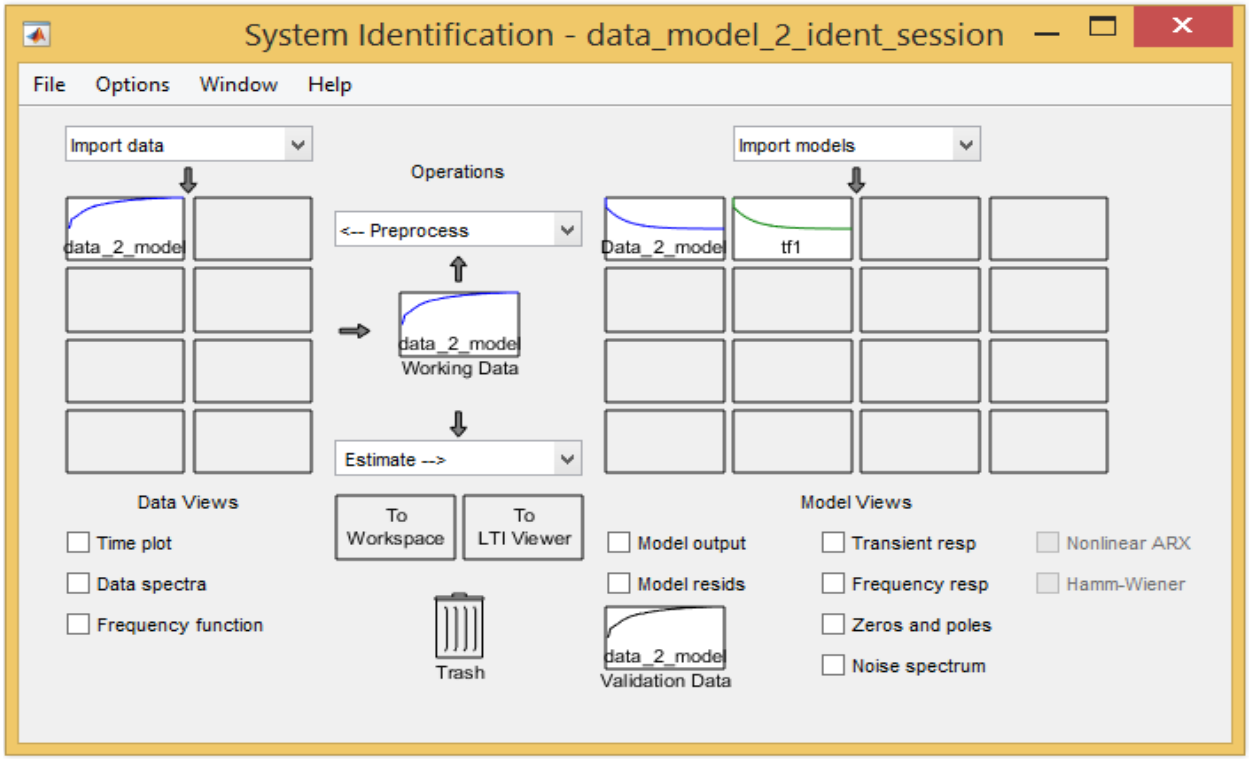


Figure 2.11. Interface of MATLAB system identification toolbox App [52].

MATLAB/Simulink provides system identification toolbox App for constructing mathematical models of dynamic systems from measured input-output data, see Figure 2.11. It lets

one create and use models of dynamic systems not easily modeled from first principles or specifications. One can use time-domain and frequency-domain input-output data to identify continuous-time and discrete-time transfer functions, process models, and state-space models. The toolbox also provides algorithms for embedded online parameter estimation [52].

The toolbox provides identification techniques such as maximum likelihood, prediction-error minimization (PEM), and subspace system identification [52]. To represent nonlinear system dynamics, one can estimate Hammerstein-Weiner models and nonlinear ARX (AutoRegressive eXogenous) models with wavelet network, tree-partition, and sigmoid network nonlinearities. The toolbox performs grey-box system identification for estimating parameters of a user-defined model. One can use the identified model for system response prediction and plant modeling in Simulink. The toolbox also supports time-series data modeling and time-series forecasting. MATLAB command *ident* or *systemIdentification* is used to access the system identification toolbox App [52].

2.4 Transient Response

A transient response or natural response is the response of a system to a change from equilibrium [53]. Example, the step response is transient response to a step input. System response can be classified as one of three types of damping that describes the output in relation to the steady-state response [53]: (1) An underdamped response is one that oscillates within a decaying envelope. The more underdamped the system, the more oscillations and longer it takes to reach steady-state. Here, the damping ratio, ξ, is always less than 1.0. (2) A critically damped response, $\xi = 1.0$, is the response that reaches the steady-state value the fastest without being underdamped. It is related to critical points in the sense that it straddles the boundary of underdamped and overdamped responses. There should be no oscillation about the steady-state

value in the ideal case. (3) An overdamped response, $\xi > 1.0$, is the response that does not oscillate about the steady-state value, but takes longer to reach than the critically damped case.

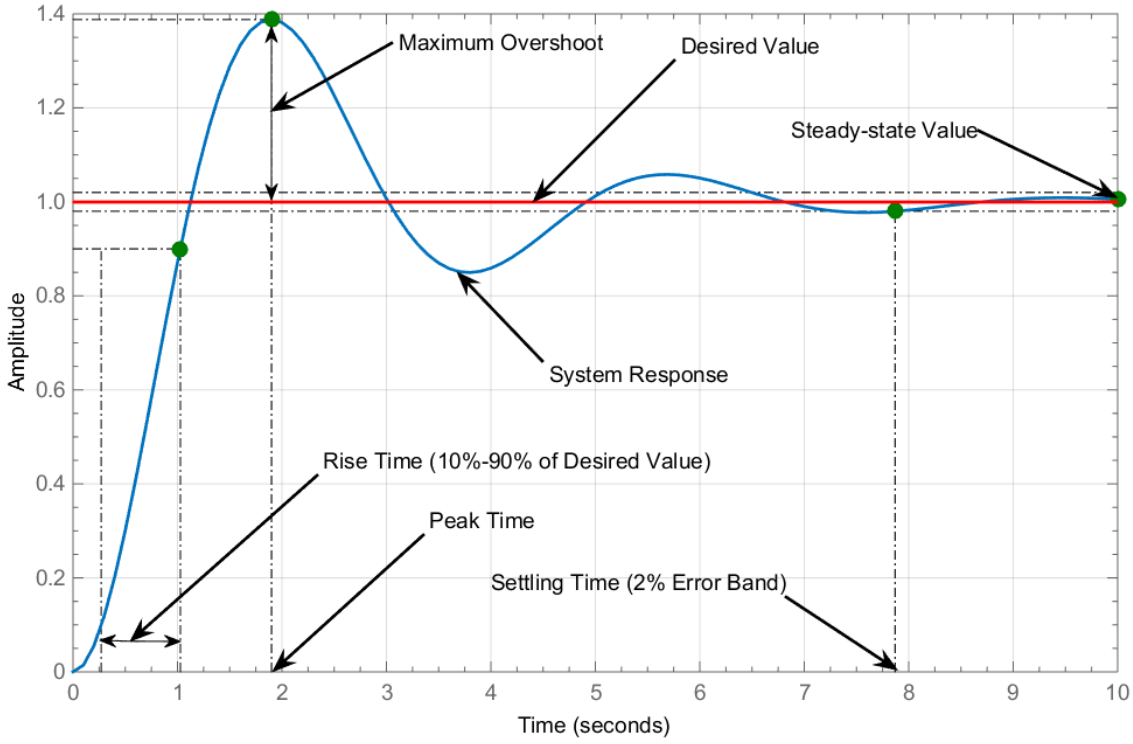


Figure 2.12. Example of underdamped system response showing the transient properties.

2.4.1 Transient response properties. A system response has the following properties (see Figure 2.12) [53]: (1) *Rise time*, t_r refers to the time required for a signal to change from a specified low value to a specified high value. Typically, these values are 10% to 90% of the step height, (2) *Maximum overshoot*, M_o refers to an output signal exceeding its final, steady-state value. It is often associated with ringing, (3) *Delay time*, t_d is the time required for the response to reach half the final value the very first time, (4) *Peak time*, t_p is the time required for the response to reach the first peak of the overshoot, (5) *Settling time*, t_s is the time elapsed from the application of an ideal instantaneous step input to the time at which the output has entered and remained within a specified error band (typically, 2% error band), and (6) *Steady-state error*, E_{ss} is the difference between the desired final output and the actual one when the system reaches a steady state, when its behavior

may be expected to continue if the system is undisturbed. The transient properties for oscillatory response such as M_o, t_p, and t_s, are related to ξ and ω_n as [54]

$$M_o = 100e^{\left(\frac{-\xi\pi}{\sqrt{(1-\xi^2)}}\right)}$$
$$t_p = \frac{\pi}{\omega_n\sqrt{(1-\xi^2)}} \tag{2.15}$$
$$t_s = \frac{4}{\xi\omega_n}$$

where ω_n is the natural frequency of the system.

2.5 Saturation, Time Delay, and System Uncertainty: Aerial Robot (Quadrotor)

Problems of robust stability of linear dynamical systems have attracted a good deal of attention in control theory during the last thirty years [55]. Initially, the model to be developed for the control system design will not take into account any disturbances or uncertainties. To test the robustness of the model in more complex situations, aerodynamic effects, uncertainties in the system parameters, measurement noises, and time delay in the control system will have to be considered; these introduce nonlinear effects on the system dynamics. The effects to be considered are saturation on the rotor angular speeds, uncertainties in the system parameters (e.g., disturbances rejection: quadrotor payload), and total time delay in the control system.

2.5.1 Saturation. The angular speed, control command, of each rotor to be sent to the quadrotor is limited, therefore the change in the total thrust or torque to be generated to produce motions is constrained. Thus, the issue of control input saturation will be studied and addressed.

2.5.2 Time delay. Estimating and analyzing time delays in dynamic systems is an important issue in many areas. Estimating delays has been an area of great research interest and has plenty of applications in fields as diverse as radar, sonar, seismology, geophysics, ultrasonic, controls, and communications for detecting, identifying, and localizing radiating sources [56].

Estimating delays is especial challenging problem [57]. Although considerable efforts have been made on parameter estimation, there are still many open problems in time-delay identification due to difficulty in formulation [58-60].

One of the challenges in designing effective control systems for autonomous control of mobile robots is existence of signal transmission delay, which has nonlinear effects on the flight performance. A controller designed using a non-delay system model may result in disappointingly slow and oscillating response due to the delays. In general, the effects of delays in closed-loop feedback systems resemble the effects of lowering the sampling frequency. The controller is forced to make use of "old" information (information about the output at some time in the past, rather than in the present) in determining the output it supplies to the plant.

When estimating time delay, the objective can be either of the following two [61]: (1) Estimate the best approximation time delay, i.e. the time-delay estimate that gives the "best" model approximation of the true system. What is "best" depends on the intended use of the model and can be measured in many different ways. For example, in automatic control, the time-delay estimate can be a means to achieve a good model in the frequency band relevant to the control, i.e. around the cross-over frequency. In the apparent time delay (the delay resulting from identifying a first order model with time delay from the data) is used for control performance monitoring of PID control loops. (2) Estimate the true time delay. This is the case in "pure time-delay" estimation, diagnosis, radar range estimation, direction of arrival estimation with array antennas, measuring blood velocity, averaging of measured signals, etc.

2.5.2.1 Time delay estimation problem. A general linear time delay estimation (TDE) problem in automatic control and signal processing systems can be formulated as [61]

$$x_1(t) = G_{p_1}u(t) + n_1(t) \qquad (2.16a)$$

$$x_2(t) = G_{p_2}u(t) + n_2(t) = G_{r_2}u(t - T_d) + n_2(t) \quad (2.16b)$$

where $G_{p_1}$and G_{r_2} are transfer functions of linear dynamic system, SISO LTI, without time delay, G_{p_2} is the transfer function with time delay, $x_1(t)$ and $x_2(t)$ are measured signals, $u(t)$ is the control input signal, $n_1(t)$ and $n_2(t)$ are measurement noise, and T_d is the delay time in seconds. The signals can be either wideband or narrowband. The signals can be either real valued or complex valued. Complex (or analytic) signal representation is often used for narrowband signals but can also be used for wideband signals [61]. The impulse responses $g_1(t)$ and $g_2(t)$ of $G_{p_1}$and G_{r_2} respectively, can also be complex valued. Complex signals and impulse responses are commonly used for bandpass systems, e.g. in radar and communications [61]. Some special cases of the general problem in (2.16) are [61]:

(1) With the noise $n_1(t) = 0$ and $G_{p_1} = 1$, we have the *active* TDE problem as (we rename x_2 to x and n_2 to n)

$$x(t) = G_r u(t - T_d) + n(t) \quad (2.17)$$

This occurs in system identification, which is useful for automatic control and range estimation in radar, etc.

(2) With the noise $n_1(t) \neq 0$ and $u(t)$ unknown we have the *passive* TDE problem. This case happens when a signal $u(t)$ has traveled two different paths and are measured with two sensors, e.g. in localization of radio sources by *Time Delay of Arrival (TDOA)* or beamforming of audio signals from an array of microphones in a car.

2.5.2.2 Time delay systems. The transfer function for a system containing time delay is given by $e^{-T_d s}$ (continuous-time) or z^{-T_d} (discrete-time), which is termed a pure time delay. If the plant model has dynamics, then the transfer function for the linear system is given in the form $G_p(s) = G_r(s)e^{-sT_d}$ (continuous-time) or $G_p(z) = G_r(z)z^{-T_d}$ (discrete-time) where $G_r(s)$ and

$G_r(z)$ are transfer functions without time delay. The following are some properties for linear time-delay systems [61]:

- A pure time-delay $G_p(s) = e^{-sT_d}$ is a linear and all-pass system.
- A continuous-time time-delay system is of infinite dimension since an infinite number of values are needed to describe the state of the system at each point of time.
- A continuous-time time-delay system in state space form can be described by a system of differential-difference equations, i.e. combined differential and difference equations.
- The transfer function $G_p(s) = e^{-sT_d}$ of a continuous-time time-delay system is not a rational function of s. $G_p(s)$ has an infinite number of poles, which is consistent with the system's infinite dimensionality.
- If the sampling period is constant and the delays are integral multiples of the sampling period, then a discrete-time time-delay system in state space form can be described by a system of pure difference equations. Such systems will be of finite dimension.
- On the other hand, if the sampling period is not constant, then a discrete-time time-delay system cannot be described by pure difference equations. Differential-difference equations are needed.
- The transfer function $G_p(z)$ of a discrete-time time-delay system is a rational function of z. $G_p(z)$ has a finite number of poles, which is consistent with the system's finite dimensionality.

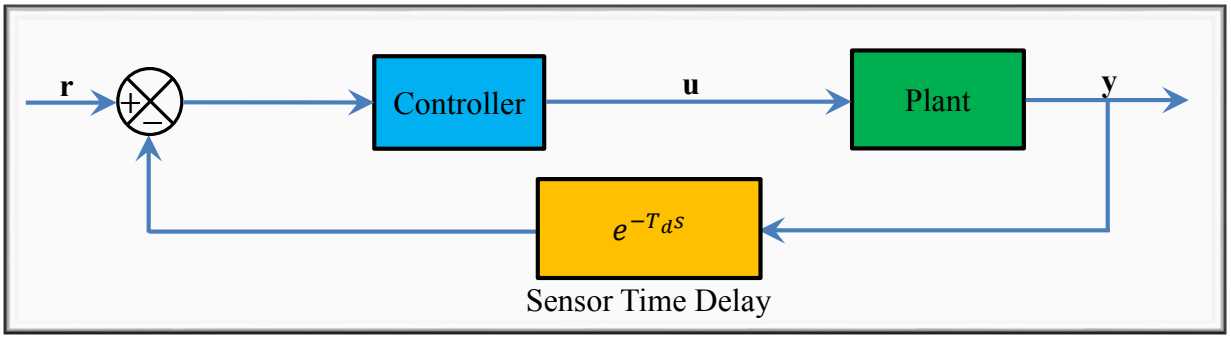


Figure 2.13. Control system showing the application of sensor time delay.

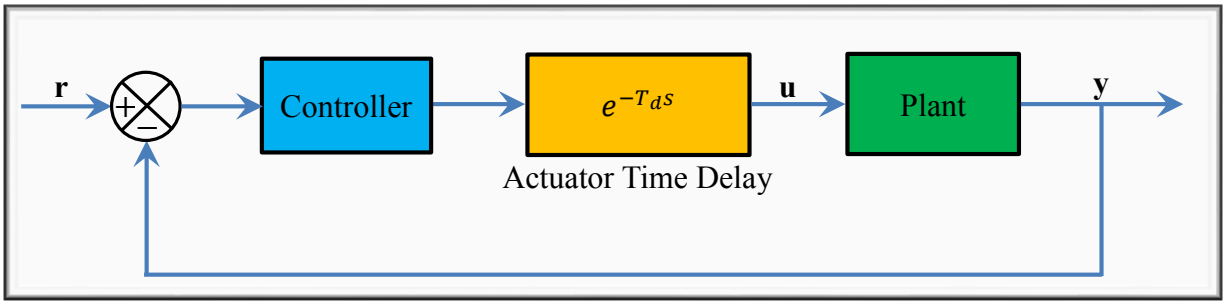


Figure 2.14. Control system showing the application of actuator time delay.

A hybrid, or mixed system, consists of a continuous-time part and a discrete-time part. An important example is a sampled continuous-time system [61]. A system with a cascade controller and unity feedback, but using an output sensor that is T_d seconds late in reporting the output (a sensor delay) would have a transfer function calculated from Figure 2.13. If the delay occurred in transmitting the output of the controller to the plant (an actuator delay), see Figure 2.14 for the illustration. For an active TDE problem, the impulse response $g_r(t)$ of G_r is illustrated in Figure 2.15.

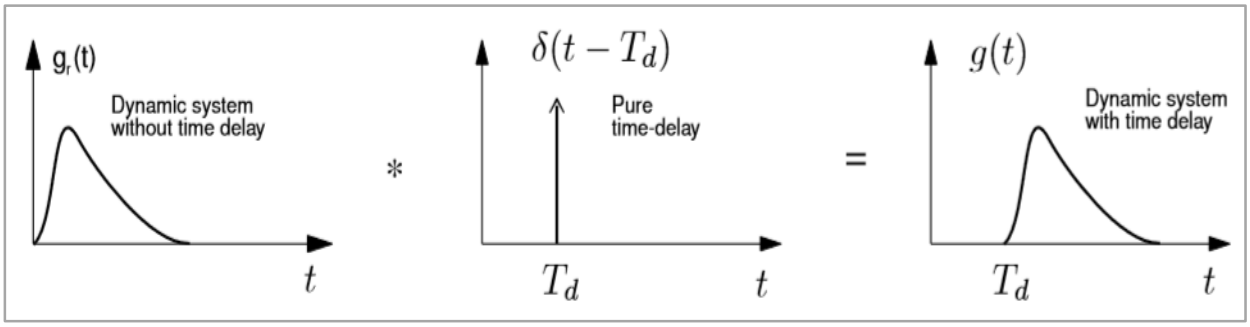


Figure 2.15. The creation of impulse response of a system with time delay [61].

This research focus on an active TDE problem, of the retarded and neutral types. The stability of linear delay systems of retarded or neutral type is a field of intense research. A major difficulty lies in the fact that the delays are usually unknown [62]. The following are special types of time-delay systems in state-space form [37]:

(a) a pure delay system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t - T_d) \tag{2.18}$$

(b) a delay system of the retarded type

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{x}(t - T_d) \tag{2.19}$$

(c) a neutral system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{C}\dot{\mathbf{x}}(t - T_d) \tag{2.20}$$

(d) a multiple delay system

$$\dot{\mathbf{x}}(t) = \sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}\left(t - T_{d_i}\right) \tag{2.21}$$

2.5.2.3 Time delay estimation methods. It is considered an advantage if a method can estimate time delays that also consist of fractions of the sampling interval [61]. However, some methods can only estimate time delays that are multiple of the sampling interval. Sometimes such methods can be used to initialize other more "free" methods. TDE has been studied in literature for a long time, especially for pure time-delay systems, but also for systems with dynamics [61]. However, there is still no clear agreement on which method is "best" for systems with dynamics. Most methods that have been suggested for active TDE (both in control and signal processing) can be put into one of the following classes [61]:

2.5.2.3.1 Time-delay approximation methods. The input and output signals are represented in a certain basis and the time delay is estimated from an approximation of the relation (a model) between the signals in this basis. There are two independent steps in these methods: (a) Estimate the approximation model. (b) Estimate the time delay from the model. The time delay is not an explicit parameter in the model. Depending on the basis, there are several subclasses, which

include time domain approximation methods, frequency domain approximation methods, and Laguerre domain approximation methods.

2.5.2.3.2 Explicit time-delay parameter methods. The time delay is an explicit parameter in the model. Approaches include one-step explicit methods, two-step explicit methods, and sampling methods. For the one-step explicit methods, termed *idproc* methods in the process industry, two variants are possible: (a) Model and estimate the time delay as a continuous parameter in a continuous-time model. Here the time delay is not restricted to be a multiple of the sampling interval, but can be a subsample time delay. (b) Model and estimate the time delay as a discrete parameter in a discrete-time model. The process models could be: (i) a first-order system with time delay, (ii) a second-order system with real poles and time delay, (iii) a second-order system with complex poles and time delay, (iv) a third-order system with real poles and time delay, or (v) a third-order system with two complex poles, one real pole and time delay.

2.5.2.3.3 Area and moment methods. These methods utilize relations between the time delay and certain areas above or below the step response and certain moments of the impulse response. There are two independent steps: a) Estimate or measure the step or impulse response. b) Estimate the time delay from these responses.

2.5.2.3.4 Higher-order statistics (HOS) methods. Their main advantage is that noise with a symmetric probability density function (PDF), e.g. Gaussian, theoretically can be removed completely by HOS. If the desired signal has a symmetric PDF, it will disappear as well.

Methods for the passive TDE problem should also be possible to use for active TDE problems. For example, if $n_1(t) = 0$ we have active TDE problem. Active TDE is thus a special case of passive TDE. The opposite, of using active TDE methods for passive TDE problems, could also be possible.

2.5.2.4 Lambert W function. In mathematics, the Lambert W function, also called the omega function or product logarithm, is a set of functions, namely the branches of the inverse relation of the function shown in equation (2.22), where W is any complex number [63].

$$\mathrm{f(W)} = W(x)e^{W(x)} = x \tag{2.22}$$

Based on the Lambert W function, a new approach has been developed for systematic analysis of control systems with time delays [64]. The function allows handling of the exponential term in the characteristic equation of Delay Differential Equations, DDEs, thanks to its unique definition in (2.22). The branches of the Lambert W function and the specific range of each branch are helpful in this new approach [64]. As seen in Figure 2.16, when the argument of the function, x, is greater than $-1/e$ the values of $W_o(x)$ is real, however, if $x < -1/e$, the values of $W_o(x)$ is complex.

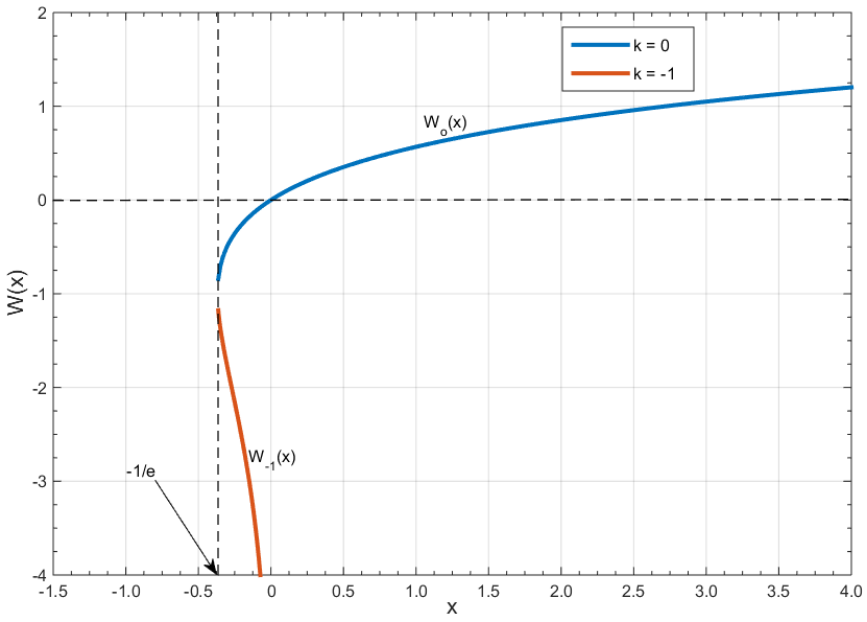


Figure 2.16. Two main branches of the Lambert W function [63].

2.5.3 System uncertainty. The extent to which a plant model is unknown will be called *uncertainty*. This is another important issue which designers might need to face. It is known that some uncertainty is always present, both in the environment of the system and in the system itself [65]. We do not know in advance exactly what disturbance and noise signals the system will be

subjected to. In the system itself, we know that no mathematical expressions can exactly model a physical system. The uncertainty may be caused by parameter changes, neglected dynamics (especially, high-frequency unmodeled dynamics), or other unspecified effects, which might adversely affect the performance of a control system [65]. Stability and performance robustness are two important issues that should be considered in control design. Generally, the form of the plant uncertainty can be parametric, nonparametric, or both [65].

In designing a controller for quadrotor helicopters, there are several important considerations, and uncertainty is one of them. Uncertainty in quadrotor systems occurs due to unmodeled factors such as friction and backlash in the mechanical parts of the DC motors, error in the machinery, which can lead to actuator degradation and failure, causing the rotor's thrust to not fully counteract each other (loss-of-thrust anomaly). System parameters such as mass, thrust constant, torque constant, and even gravitational strength (e.g., of external disturbances) can vary due to some of the factors mentioned, and also depend on where the device is being used. The time delay due to processing or communication in the control system is also an inherent and potential uncertainty [25]. These problems are amplified in the case of actuator failures, where the quadrotor has lost some of its control effectiveness [25].

During the design process, the engineer may want to consider the extent to which changes in system parameters affect the behavior of a system [65]. One of the main advantages of feedback is that it can be used to make the response of a system relatively independent of certain types of changes or inaccuracies in the plant model [65]. Ideally, parameter changes due to heat, humidity, age, or other causes mentioned above should not appreciable affect a system's performance [65]. The degree to which changes in system parameters affect system transfer functions, and hence performance, is called *sensitivity*. The greater the sensitivity, the worse is the effect of a parameter

change. Adaptive control (e.g., MRAC) is an attractive candidate for this type of aircraft because of its ability to generate high performance tracking in the presence of parametric uncertainties [25, 65].

Systems with uncertainty is a wide class comprising systems of various nature (continuous/discrete, linear/nonlinear, etc.) characterized by the presence of unknown parameters. In these systems, the target is typically to carry out analysis and synthesis tasks for a family of admissible values of the uncertainty. The various possible behaviors of the system are determined by designing robust control strategies. A system with uncertainty in parameter, m_u, will have best estimate of its true value, m_u', as [66]

$$m_u' = \bar{m}_u \pm \Delta_{m_u}\,(P\%) \tag{2.23}$$

where its mean (nominal) value is $\bar{m}_u$, Δ_{m_u} is the uncertainty, and P is the probability level. If the measured variable is based on a finite sampling, then the *Student's t distribution* can be used to estimate Δ_{m_u} as $\pm t_{v,P} s_{\bar{m}_u}$, which expresses a confidence interval about $\bar{m}_u$, with coverage factor t at the assigned probability, $P\%$, within which one should expect m_u' to fall, and $v = N - 1$ is the degrees of freedom in the sample variance, $s_{m_u}^2$[66]. The standard deviation of the means, $s_{\bar{m}_u} = s_{m_u}/\sqrt{N}$, where N is total number of measurements [66].

A way to ensure stability robustness with respect to these uncertainties, is to employ stability criteria valid for any nonnegative value of the time delay [62]. In the study of these problems, the notion of *stability radius* has been proven to be an effective tool. The stability radius of an object (system, function, matrix, parameter, etc.) at a given nominal point is the radius of the largest ball, centered at the nominal point, all of whose elements satisfy pre-determined stability conditions [67].

In its simplest form, the stability radius of a given asymptotically stable, nominal, system without time delay, $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)$, is the *maximal* $r > 0$ for which all systems, perturbed, of the form

$$\dot{\mathbf{x}}(t) = (\mathbf{A} + \mathbf{E}\Delta\,\mathbf{F})\mathbf{x}(t), \qquad \|\Delta\| < r, \tag{2.24}$$

are asymptotically stable [55]. Here, Δ is the unknown disturbance or perturbation matrix, $\mathbf{E}$ and $\mathbf{F}$ are known scaling matrices defining the *structure* of the perturbations, where $\mathbf{A} \in \mathbb{C}^{nxn}, \sigma(\mathbf{A}) \subset \mathbb{C}$, $\mathbf{E} \in \mathbb{C}^{nxm}$, $\mathbf{F} \in \mathbb{C}^{pxn}$, $\Delta \in \mathbb{C}^{mxp}$, $\mathbb{C}$ is the complex plane, and σ is singular values [55, 68]. The matrices $\mathbf{E}$ and $\mathbf{F}$ may reflect, for instance, the possibility that not all of the elements of $\mathbf{A}$ are subject to perturbations. In this case, the system matrix, $\mathbf{A}$, is subjected to *structured perturbations*, denoted as [55]

$$\mathbf{A} \to \mathbf{A} + \mathbf{E}\Delta\,\mathbf{F} \tag{2.25}$$

Depending upon whether complex or real disturbances, Δ, are considered, r is called complex or real stability radius, respectively [55]. It is important to note that these two stability radii are in general distinct. The analysis and computation of r is much simpler for a class of positive systems, a class of systems that has the important property that its state variables are never negative, given a positive initial state [69]. It has been shown that if $\mathbf{A}$ is a Metzler matrix (i.e., all off-diagonal entries of $\mathbf{A}$ are nonnegative) and $\mathbf{E}$ and $\mathbf{F}$ are nonnegative matrices, then the two stability radii coincide [55]. It is worth noticing that the notion of stability radius can be extended to various perturbation types, and among perturbation types, the two most well known in control theory are [55]

$$\mathbf{A} \to \mathbf{A} + \sum_{i=1}^{N} \mathbf{E}_i \Delta_i \mathbf{F}_i \qquad \text{(multi perturbation)} \tag{2.26}$$

$$\mathbf{A} \rightarrow \mathbf{A} + \sum_{i=1}^{N} \boldsymbol{\delta}_i \mathbf{B}_i \qquad \text{(affine perturbation)} \tag{2.27}$$

This research work make use of the work done by [37], which present readily computable formulae with respect to an arbitrary stability region in the complex plane. The formulae computes the real stability radii for perturbed LTI time-delay control systems. The real stability radius, $r_{\Re}$, is determined for a perturbed time-delayed P control system, retarded type, given as [37]

$$\dot{\mathbf{x}} = (\mathbf{A} + \mathbf{E}\Delta_{\mathrm{A}}\,\mathbf{F}_{\mathrm{A}})\mathbf{x} + (\mathbf{B} + \mathbf{E}\Delta_{\mathrm{B}}\,\mathbf{F}_{\mathrm{B}})\mathbf{x}(t - T_d) \tag{2.28}$$

where Δ_{A} is the perturbed value in $\mathbf{A}$, Δ_{B} is the perturbed value in $\mathbf{B}$, $\mathbf{F}_{\mathrm{A}}$, $\mathbf{F}_{\mathrm{B}}$, and, $\mathbf{E}$ are perturbed parameters, then $r_{\Re}$ is given as [37]

$$r_{\Re}(\mathbf{A}; \mathbf{B}; \mathbf{E}, \mathbf{F}_{\mathrm{A}}, \mathbf{F}_{\mathrm{B}}; \mathbb{C}_b) = \left\{ \sup_{s \in \partial \mathbb{C}_g} \inf_{\gamma \epsilon (0,1]} \sigma_2[\mathbf{P}(\gamma)] \right\}^{-1} \tag{2.29}$$

where the matrix $\mathbf{P}(\gamma) \in \Re^{4p\mathrm{x}2m}$ is given as

$$\mathbf{P}(\gamma) = \begin{bmatrix} \Re\big(\mathbf{W}(s)\big) & -\gamma\,\Im\big(\mathbf{W}(s)\big) \\ \gamma^{-1}\,\Im\big(\mathbf{W}(s)\big) & \Re\big(\mathbf{W}(s)\big) \end{bmatrix} \tag{2.30}$$

where $\mathbb{C} = \mathbb{C}_g \cup \mathbb{C}_b$, such that $\mathbb{C}_g$ is the open left complex plane, $\partial \mathbb{C}_g$ denote the boundary of $\mathbb{C}_g$, $s = j\omega$ is the Laplace operator, ω is the system natural frequency, $\mathbf{W} \in \mathbb{C}^{pxm}$ is a complex matrix, whose singular values, ordered non-increasing, are denoted by $\sigma_i(\mathbf{W}), i = 1, \ldots, min\{p, m\}$ [70]. The unperturbed system from (2.28) is stable if its eigenvalues are contained in $\mathbb{C}_g$ [37].

The function, $\mathbf{P}(\gamma)$, to be minimized is a unimodal function on the search range, $\gamma \epsilon (0, 1]$, i.e., any local minimum is a global minimum [70]. The upper bound or the critical frequency for ω, denoted ω^*, on the global maximizer in (2.29), is computable at a small cost compared to that of performing the global maximization [37]. The corresponding values for s and γ are denoted as s^* and γ^*, respectively. At ω^*, the minimum of the second singular value, σ_2, of $\mathbf{P}(\gamma)$ occurs at

$\gamma = \gamma^*$. The destabilizing, worst, perturbation matrices with minimum norm, $\|\Delta\| = \|[\Delta_A, \Delta_B]\| = r_{\Re}$, i.e., the smallest real Δ is given by [37, 70].

$$\Delta = r_{\Re}[\mathbf{v}_1 \ \mathbf{v}_2][\mathbf{u}_1 \ \mathbf{u}_2]^{TH} \tag{2.31}$$

where $[\mathbf{v}_1 \ \mathbf{v}_2]$ and $[\mathbf{u}_1 \ \mathbf{u}_2]$ are the left and right singular vectors, respectively, of $\mathbf{P}(\gamma)$ in (2.30).

Many standard search algorithms, such as golden section search, can be used with guaranteed convergence to a global minimum [70]. For this research work, MATLAB function, *fminbnd*, is used for the function minimization. It determines the minimum of single-variable function on a fixed interval. For example, $q = fminbnd(fun, q_1, q_2)$ returns a value q that is a local minimizer of the function that is described in *fun* in the interval $q_1 < q < q_2$, where *fun* is a function handle.

2.5.3.1 Payload. Payload is the carrying capacity of an aircraft or launch vehicle, usually measured in terms of weight [71]. Depending on the nature of the flight or mission, the payload of a vehicle may include cargo, passengers, flight crew, munitions, scientific instruments or experiments, or other equipment. Extra fuel, when optionally carried, is also considered part of the payload. In a commercial context (i.e., an airline or air freight carrier), payload may refer only to revenue-generating cargo or paying passengers [71].

For a rocket, the payload can be a satellite, space probe, or spacecraft carrying humans, animals, or cargo. For a ballistic missile, the payload is one or more warheads and related systems; the total weight of these systems is referred to as the *throw-weight* [71]. The fraction of payload to the total liftoff weight of the air or spacecraft is known as the *payload fraction*. When the weight of the payload and fuel are considered together, it is known as the *useful load fraction*. In spacecraft, *mass fraction* is normally used, which is the ratio of payload to everything else, including the rocket structure [71].

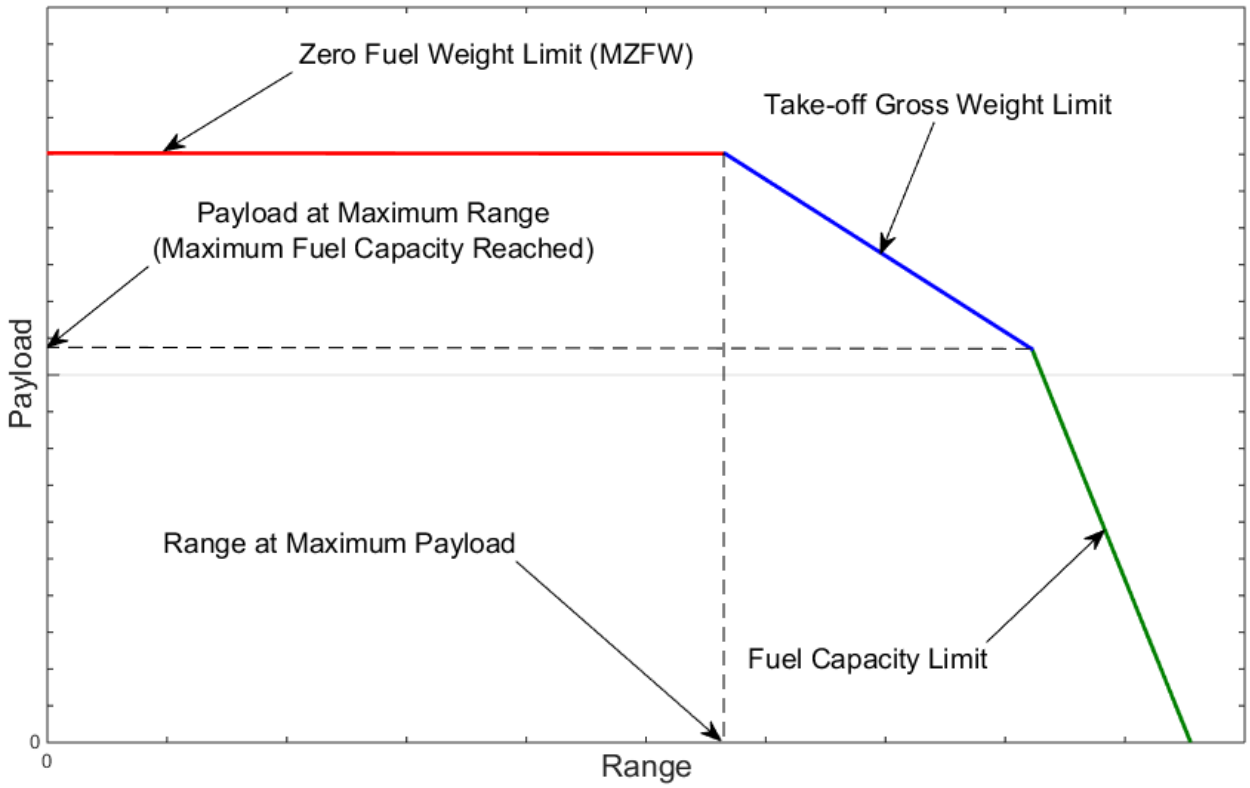


Figure 2.17. Trade-off between payload and range of an aircraft [71].

There is a natural trade-off between the payload and the range of an aircraft. A *payload range diagram* (also known as the *elbow chart*) illustrates the trade-off, see Figure 2.17 [71]. The top horizontal line represents the *maximum payload*. It is limited structurally by *maximum zero-fuel weight (MZFW)* of the aircraft. Maximum payload is the difference between MZFW and *operational empty weight (OEW)*. Moving left-to-right along the line shows the constant maximum payload as the range increases. More fuel needs to be added for more range [71].

The vertical line represents the range at which the combined weight of the aircraft, maximum payload and needed fuel reaches the *maximum take-off weight (MTOW)* of the aircraft. If the range is increased beyond that point, payload has to be sacrificed for fuel [71]. The MTOW is limited by a combination of the maximum net power of the engines and the lift/drag ratio of the wings. The diagonal line after the range-at-maximum-payload point shows how reducing the

payload allows increasing the fuel (and range) when taking off with the MTOW. The second kink in the curve represents the point at which the maximum fuel capacity is reached. Flying further than that point means that the payload has to be reduced further, for an even lesser increase in range. The *absolute range* is thus the range at which an aircraft can fly with maximum possible fuel without carrying any payload [71].

Safe, cooperative transportation of possibly large or bulky payloads by quadrotor is extremely important in various missions, such as military operations, search and rescue, Mars surface explorations, and personal assistants. There is growing interest in the research community towards the development of aerial robotic systems capable of acquiring external payloads or otherwise physically interacting with objects in the environment [72].

The application of rotorcraft to autonomous load carrying and transport is a new frontier for UAVs, and there are a number of substantial challenges and open research questions related to this application. This task requires that hovering vehicles remain stable and balanced in flight as payload mass is added to the vehicle. If payload is not loaded centered or the vehicle properly trimmed for offset loads, the robot will experience bias forces that must be rejected [72].

CHAPTER 3

Methodologies, Algorithms, and Modeling

This chapter discusses the kinematics and dynamics of the ground and aerial robots. It also presents the control algorithms and modeling that were used for the research work.

3.1 Ground Robot: X80SV

This section introduces the kinematic model of a DDWMR. Also, the kinematic model of a unicycle is also introduced. The section continues with the control algorithms built for the different WMR behaviors such as 'move to a point', 'move to a pose', 'follow a line', 'follow a circle', and 'avoid obstacles'. The section ends with control navigation system that combines 'go-to-goal', 'follow-wall', and 'avoid obstacles'.

3.1.1 Kinematic modeling of DDWMR. The DDWMR setup used for the presented study is shown in Figure 3.1 (top view). The mobile robot is made up of a rigid body and non-deforming wheels, and it is assumed that the vehicle moves on a plane without slipping, i.e., there is a pure rolling contact between the wheels and the ground.

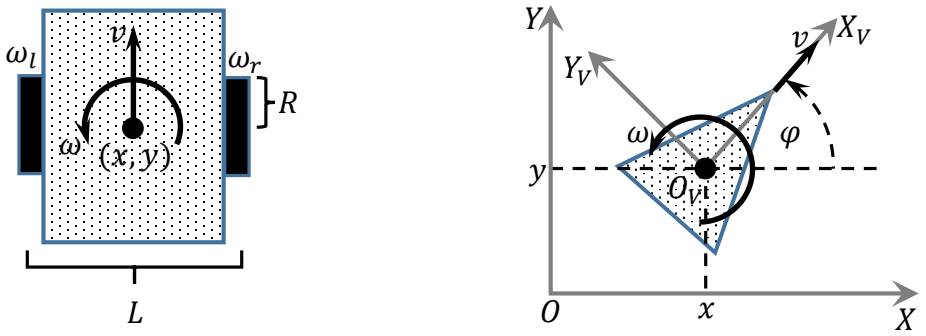


Figure 3.1. Coordinates for differential drive wheeled mobile robot (DDWMR) [73].

The configuration of the vehicle is represented by the generalized coordinates $q = (x, y, \varphi)$, where (x, y) is the position and φ is the orientation (heading or yaw) of the center of the axis of the wheels, C, with respect to a global inertial frame, $\{O, X, Y\}$. Let $\{O_V, X_V, Y_V\}$ be the vehicle

frame. The vehicle's velocity is by definition v in the vehicle's x-direction, L is distance between the wheels, R is radius of the wheels, ω_r is the right wheel angular velocity, ω_l is the left wheel angular velocity, and ω is the heading rate. The kinematic model of the DDWMR based on the stated coordinate is given as [73]

$$\begin{aligned} \dot{x} &= \frac{R}{2}(\omega_r + \omega_l)\cos\varphi \\ \dot{y} &= \frac{R}{2}(\omega_r + \omega_l)\sin\varphi \\ \dot{\varphi} &= \frac{R}{L}(\omega_r - \omega_l) \end{aligned} \tag{3.1}$$

For the purpose of implementation the kinematic model of a unicycle is used, which corresponds to a single upright wheel rolling on the plane, with the equation of motion given as [73]

$$\begin{aligned} \dot{x} &= v\cos\varphi \\ \dot{y} &= v\sin\varphi \\ \dot{\varphi} &= \omega \end{aligned} \tag{3.2}$$

The inputs in (3.1) and (3.2) are $\omega_r, \omega_l, v,$ and ω. These inputs are related as

$$\begin{aligned} \omega_r &= \frac{2v + \omega L}{2R} \\ \omega_l &= \frac{2v - \omega L}{2R} \end{aligned} \tag{3.3}$$

3.1.2 Control algorithms. Control of the unicycle model inputs, **u**, is about calculating the appropriate input values by applying the traditional PID-feedback controller shown in (2.4), given as

$$\mathbf{u} = (v\ \omega)^T = PID(\boldsymbol{e}) \tag{3.4}$$

If the vehicle is driven at a constant linear velocity, $v = v_o$, then only ω will vary with time.

3.1.2.1 Developing individual controllers. This section presents control algorithms that make mobile robots 'move to a point', 'move to a pose', 'follow a line', 'follow a circle', and 'avoid obstacles'.

3.1.2.1.1 Moving to point. Consider a robot moving toward a goal point, (x_g, y_g), from a current position, (x, y), in the xy-plane, as depicted in Figure 3.2.

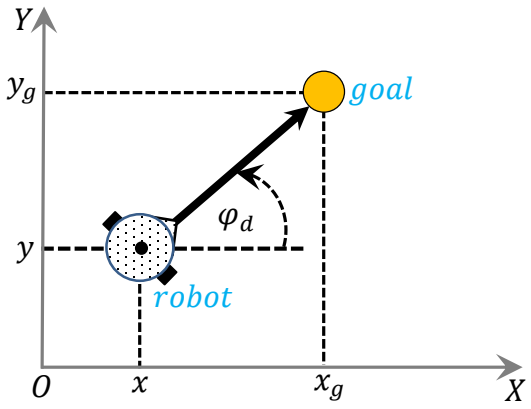


Figure 3.2. Coordinates for moving to a point.

The desired heading (robot's relative angle), φ_g, is determined as [73]

$$\varphi_g = \varphi_{goal} = \arctan\left(\frac{y_g - y}{x_g - x}\right) \tag{3.5}$$

and the error, e, is defined as

$$e = \varphi_g - \varphi \tag{3.6}$$

To ensure $e \in [-\pi, \pi]$, a corrected error, e', is used instead of e, computed as [73]

$$e' = arctan2\big(sin(e), cos(e)\big) \in [-\pi, \pi] \tag{3.7}$$

Thus, ω can be controlled using

$$\omega = K_h e' \tag{3.8}$$

where K_h a proportional controller gain. If the robot's linear velocity is to be controlled, a proportional controller gain, K_v, is applied to the distance from the goal, computed as [39]

$$v = K_v \sqrt{(x_g - x)^2 + (y_g - y)^2} \tag{3.9}$$

3.1.2.1.2 Moving to pose. The above controller could drive the robot to a goal position, but the final orientation depends on the starting position. In order to control the final orientation (3.2) is rewritten in matrix form as

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\varphi}\end{pmatrix}=\begin{pmatrix}cos\varphi & 0\\ sin\varphi & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}v\\ \omega\end{pmatrix} \tag{3.10}$$

(3.10) is then transformed into the polar coordinate form using the notation shown in Figure 3.3.

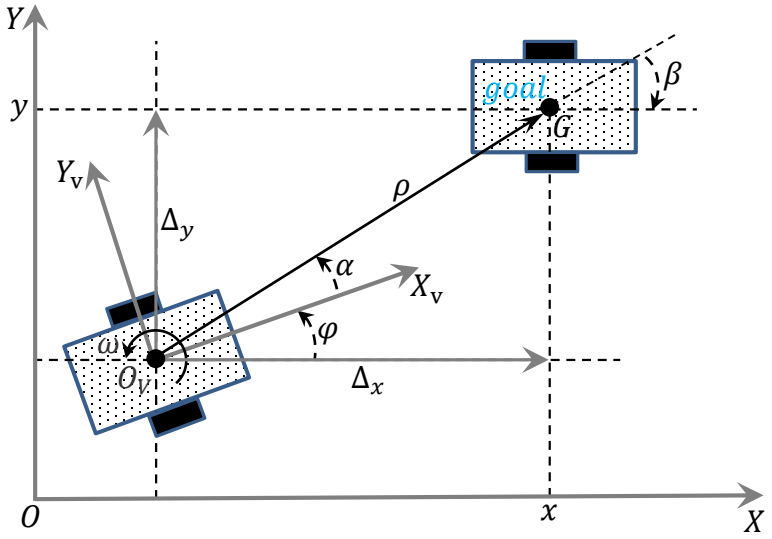


Figure 3.3. Coordinates for moving to a pose [39].

Applying a change of variables, we obtain [39]

$$\begin{aligned}\rho &= \sqrt{{\Delta_x}^2+{\Delta_y}^2}\\ \alpha &= \arctan\left(\frac{\Delta_y}{\Delta_x}\right)-\varphi\\ \beta &= -\varphi-\alpha\end{aligned} \tag{3.11}$$

which results in [39]

$$\begin{pmatrix}\dot{\rho}\\ \dot{\alpha}\\ \dot{\beta}\end{pmatrix}=\begin{pmatrix}-cos\alpha & 0\\ \frac{sin\alpha}{\rho} & -1\\ -\frac{sin\alpha}{\rho} & 0\end{pmatrix}\begin{pmatrix}v\\ \omega\end{pmatrix},\ \ if\ \alpha\in\left(-\frac{\pi}{2},\frac{\pi}{2}\right] \tag{3.12}$$

and this assumes the goal, G, is in front of the vehicle. The linear control law [39]

$$v = K_\rho \rho$$
$$\omega = K_\alpha \alpha + K_\beta \beta \tag{3.13}$$

drives the robot to unique equilibrium at $(\rho, \alpha, \beta) = (0,0,0)$. The intuition behind this controller is that the terms $K_\rho \rho$ and $K_\alpha \alpha$ drive the robot along a line toward G while the term $K_\beta \beta$ rotates the line so that $\beta \to 0$ [39]. The closed-loop system

$$\begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} -K_\rho cos\alpha \\ K_\rho sin\alpha - K_\alpha \alpha - K_\beta \beta \\ -K_\rho sin\alpha \end{pmatrix} \tag{3.14}$$

is stable so long as $K_\rho > 0,\ \ K_\beta < 0,\ \ K_\alpha - K_\rho > 0$ [39]. For the case where the goal is behind the robot, that is $\alpha \notin \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$, the robot is reverse by negating v and ω in the control law. The velocity v always has a constant sign which depends on the initial value of α.

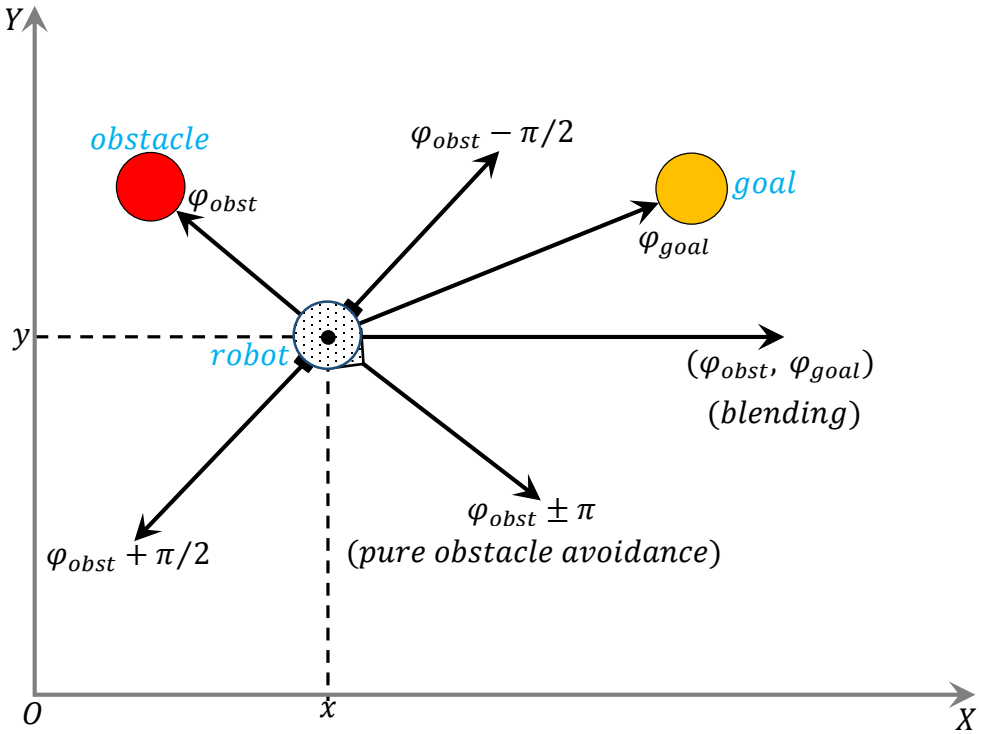


Figure 3.4. Coordinates for avoiding obstacle [73].

3.1.2.1.3 Obstacle avoidance. In a real environment robots must avoid obstacles in order to go to a goal. Depending on the positions of the goal and the obstacle(s) relative to the robot, the

robot needs move to the goal using φ_g from a 'pure go-to-goal' behavior or blending the 'avoid obstacle' and the 'go-to-goal' behaviors. In pure obstacle avoidance the robot drives away from the obstacle and move in the opposite direction. The possible heading that can be used in the control law discussed in *Section 3.1.2.1.1* are shown in Figure 3.4, where φ_{obst} is the obstacle heading.

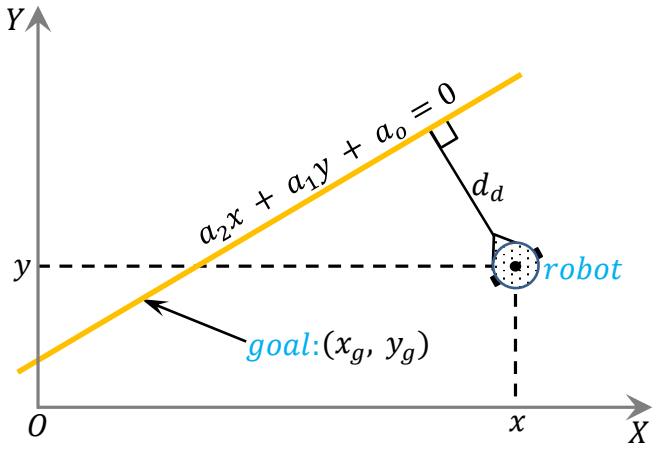


Figure 3.5. Coordinates for following a line.

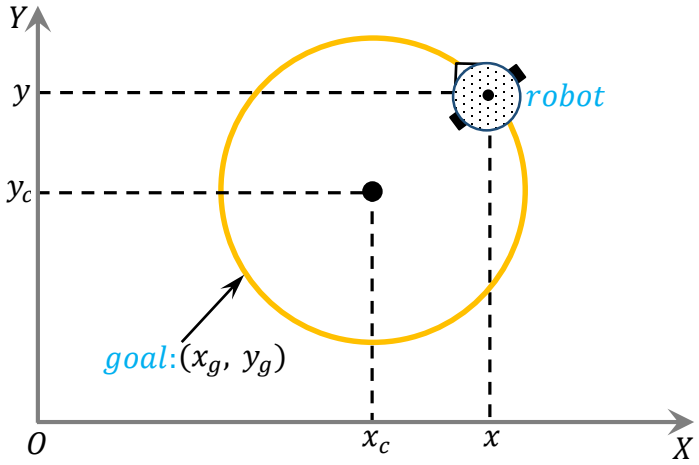


Figure 3.6. Coordinates for following a circle.

3.1.2.1.4 Following a line. Another useful task for a mobile robot is to follow a line on a plane defined by $a_2x + a_1y + a_o = 0$, as shown in Figure 3.5. This requires two controllers to

adjust the heading. One controller steers the robot to minimize the robot's normal distance from the line given by [39]

$$d_d = \frac{(a_2, a_1, a_o).(x, y, 1)}{\sqrt{a_2^2 + a_1^2}} \tag{3.15}$$

The proportional controller [39]

$$\alpha_d = -K_d d_d, \qquad K_d > 0 \tag{3.16}$$

turns the robot toward the line. The second controller adjusts the heading angle to be parallel to the line [39]

$$\varphi_g = \arctan\left(-\frac{a_2}{a_1}\right) \tag{3.17}$$

using the proportional controller [39]

$$\alpha_h = K_h(\varphi_g - \varphi), \qquad K_h > 0 \tag{3.18}$$

The combined control law [39]

$$\omega = -K_d d_d + K_h(\varphi_g - \varphi) \tag{3.19}$$

turns the wheel so as to drive the robot toward the line and moves along it.

3.1.2.1.5 Following a circle. Instead of a straight line the robot can follow a defined path on the xy-plane, and in this section the robot follows a circle, as shown in Figure 3.6. This problem is very similar to the control problem presented in *Section 3.1.2.1.1*, except that this time the point is moving. The robot maintains a distance d_l behind the pursuit point and an error, e, can be formulated as [39]

$$e = \sqrt{(x_g - x)^2 + (y_g - y)^2} - d_l \tag{3.20}$$

that will be regulated to zero by controlling the robot's velocity using a PI controller

$$v_d = PI(e) = K_P e(t) + K_I \int_0^t e(\tau) d\tau \tag{3.21}$$

The integral term is required to provide a finite velocity demand v_d when the following error is zero. The second controller steers the robot toward the target which is at the relative angle given by (3.5) and a controller given by (3.18).

3.1.2.2 Developing navigation control algorithm. This section introduces how the navigation architecture that consist of go-to-goal, follow-wall, and avoid obstacle behaviors was developed. In order to develop the navigation system, a low-level planning was used by starting with a simple model whose input can be calculated by using a PID controller or transform into actual robot input, depicted in Figure 3.7. For this simple planning a desired motion vector, **x**, is picked and set equal to the input, $\mathbf{u}_m$, see (3.22), where $\mathbf{u}_m = (u_1\ u_2)^T$ [73].

$$\dot{\mathbf{x}} = \mathbf{u}_m = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_m, \quad \mathbf{x} \in \Re^2 \tag{3.22}$$

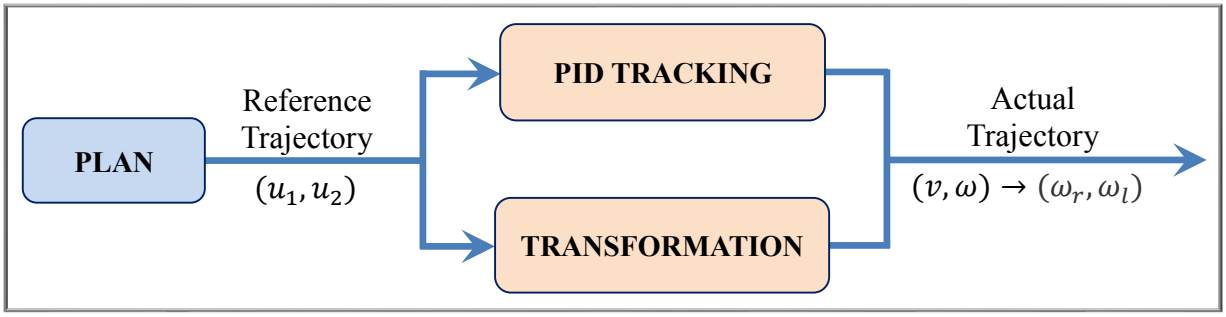


Figure 3.7. Planning model input to actual robot input [73].

This selected system is unstable, but controllable as compared to the unicycle system which is non-linear and not controllable even after it has been linearized. This layered architecture makes the DDWMR act like the point mass model shown in (3.22).

3.1.2.2.1 Go-to-goal (GTG) mode. Consider the point mass moving toward a goal point, $\mathbf{x}_g$, with current position as **x** in the xy-plane, see Figure 3.8. The error, $\mathbf{e} = \boldsymbol{x}_g - \mathbf{x}$, is controlled by the input $\mathbf{u}_m = \mathbf{Ke}$, where **K** is gain matrix. Since $\dot{\mathbf{e}} = -\mathbf{Ke}$ the system is asymptotically stable if $\mathbf{K} > \mathbf{0}$. An appropriate **K** is selected to obey the function shown in Figure 3.9a such that $\dot{\mathbf{e}} =$

$-\mathbf{K}(\|\mathbf{e}\|)\mathbf{e}$, where a and b are constants to be selected; in this way the robot will not go faster further away from the goal.

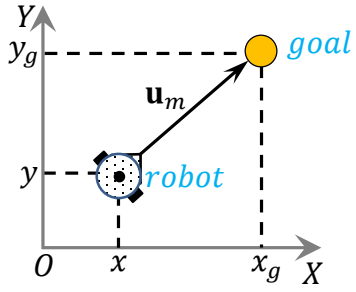


Figure 3.8. Coordinates for go-to-goal.

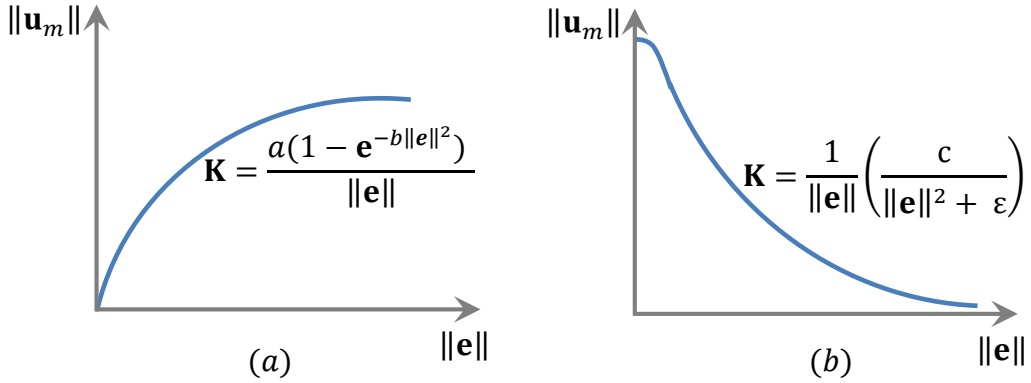


Figure 3.9. Suitable graph for **K** [73] (a) go-to-goal mode (b) obstacle avoidance mode.

3.1.2.2.2 Obstacle avoidance (AO) mode. Let the obstacle position be $\mathbf{x}_o$, then $\mathbf{e} = \mathbf{x}_o - \mathbf{x}$ is controlled by the input $\mathbf{u}_m = -\mathbf{K}\mathbf{e}$, and since $\dot{\mathbf{e}} = \mathbf{K}\mathbf{e}$ the system is desirably unstable if $\mathbf{K} > \mathbf{0}$, see Figure 3.10. An appropriate **K** is selected to obey the function shown in Figure 3.9b such that $\dot{\mathbf{e}} = \mathbf{K}(\|\mathbf{e}\|)\mathbf{e}$, where c and ε are constants to be selected.

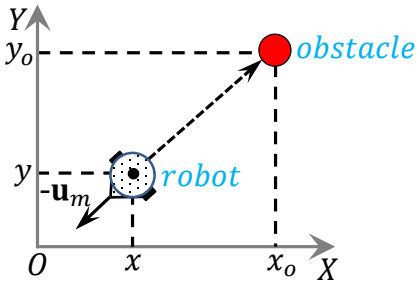


Figure 3.10. Coordinates for obstacle avoidance.

3.1.2.2.3 Blending AO and GTG mode. In a 'pure GTG' mode, $\mathbf{u}_{GTG}$, or 'pure AO' mode, $\mathbf{u}_{AO}$, or what is termed as hard switches, performance can be guaranteed but the ride can be bumpy, and the robot can encounter the *zeno phenomenon*. A control algorithm for blending the $\mathbf{u}_{GTG}$ and $\mathbf{u}_{AO}$ modes is given by [73].

$$\mathbf{u}_{GTG,AO} = \alpha(\Delta)\mathbf{u}_{GTG} + \big(1 - \alpha(\Delta)\big)\mathbf{u}_{AO}, \;\; \alpha(\Delta) \in [0\ 1] \tag{3.23}$$

where Δ is a constant distance to the obstacle/boundary and α is the blending function to be selected, giving appropriately as an exponential function by [73]

$$\alpha(\Delta) = 1 - e^{-\beta\Delta} \tag{3.24}$$

where β is a constant to be selected. This algorithm ensures a smooth ride but does not guarantee performance.

3.1.2.2.4 Follow-wall (FW) mode. As pointed out in *Section 3.1.2.2.2*, in a pure obstacle avoidance mode the robot drives away from the obstacle and move in the opposite direction, but this is overly cautious in a real environment where the task is to go to a goal. The robot should be able to avoid obstacles by going around its boundary, and this situation leads to what is termed as the follow-wall or an induced or sliding mode, $\mathbf{u}_{FW}$, between the $\mathbf{u}_{GTG}$ and $\mathbf{u}_{AO}$ modes; this is needed for the robot to negotiate complex environments.

The FW mode maintains Δ to the obstacle/boundary as if it is following it, and the robot can clearly move in two different directions, clockwise (c) and counter-clockwise (cc), along the boundary, see Figure 3.11. This is achieved by rotating $\mathbf{u}_{AO}$ by $\pi/2$ or $-\pi/2$ to obtain $\mathbf{u}_{FW}^{cc}$ or $\mathbf{u}_{FW}^{c}$ respectively, and then scaled by δ to obtain a suitable induced mode, $(3.25 - 3.27)$, where $R(\varphi)$ is a rotation matrix [73].

$$R(\varphi) = \begin{bmatrix} cos\varphi & -sin\varphi \\ sin\varphi & cos\varphi \end{bmatrix} \tag{3.25}$$

$$\mathbf{u}_{FW}^{cc} = \delta R(\pi/2)\mathbf{u}_{AO} = \delta \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{u}_{AO} \tag{3.26}$$

$$\mathbf{u}_{FW}^{c} = \delta R(-\pi/2)\mathbf{u}_{AO} = \delta \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \mathbf{u}_{AO} \tag{3.27}$$

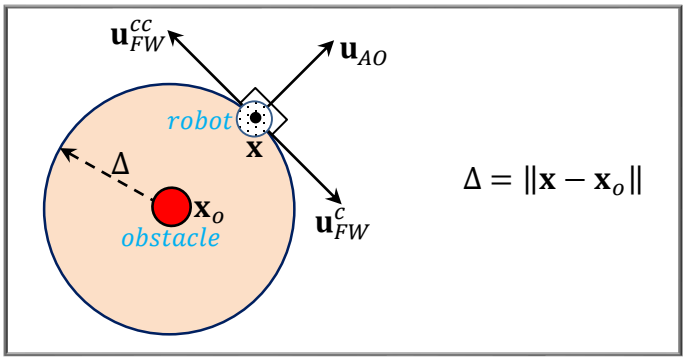


Figure 3.11. Setup for follow-wall mode [73].

The direction the robot selects to follow the boundary is determined by the direction of $\mathbf{u}_{GTG}$, and it is determined using the dot product of $\mathbf{u}_{GTG}$ and $\mathbf{u}_{FW}$, as shown in (3.28) and (3.29) [73].

$$\langle \mathbf{u}_{GTG}, \mathbf{u}_{FW}^{cc} \rangle = (\mathbf{u}_{GTG})^T \mathbf{u}_{FW}^{cc} > 0 \Rightarrow \mathbf{u}_{FW}^{cc} \tag{3.28}$$

$$\langle \mathbf{u}_{GTG}, \mathbf{u}_{FW}^{c} \rangle = (\mathbf{u}_{GTG})^T \mathbf{u}_{FW}^{c} > 0 \Rightarrow \mathbf{u}_{FW}^{c} \tag{3.29}$$

Another issue to be addressed is when the robot releases $\mathbf{u}_{FW}$, that is when to stop sliding. The robot stops sliding when "enough progress" has been made and there is a "clear shot" to the goal, as shown in (3.30 − 3.32), where τ_s is the time of last switch [73].

$$enough\ progress\colon \ \|\mathbf{x} - \mathbf{x}_g\| < d_{\tau_s} \tag{3.30}$$

$$where \quad d_{\tau_s} = \|\mathbf{x}(\tau_s) - \mathbf{x}_g\| \tag{3.31}$$

$$clear\ shot\colon \langle \mathbf{u}_{AO}, \mathbf{u}_{GTG} \rangle = (\mathbf{u}_{AO})^T \mathbf{u}_{GTG} > 0 \tag{3.32}$$

3.1.2.2.5 Implementation of navigation algorithm. The behaviors or modes discussed above are put together to form the navigation architecture shown in Figure 3.12. The robot starts at the state, $\mathbf{x}_o$, and arrives at the goal, $\mathbf{x}_g$, switching between the three different operation modes,

using *logic control*. This system of navigation is termed the *hybrid automata* where the navigation system has been described using both the continuous dynamics and the discrete switch logic. An illustration of this navigation system is shown in Figure 3.13. The robot avoids the rectangular block as it moves to the goal. It goes around the boundary clockwise since $\langle \mathbf{u}_{GTG}, \mathbf{u}_{FW}^{c} \rangle > 0$.

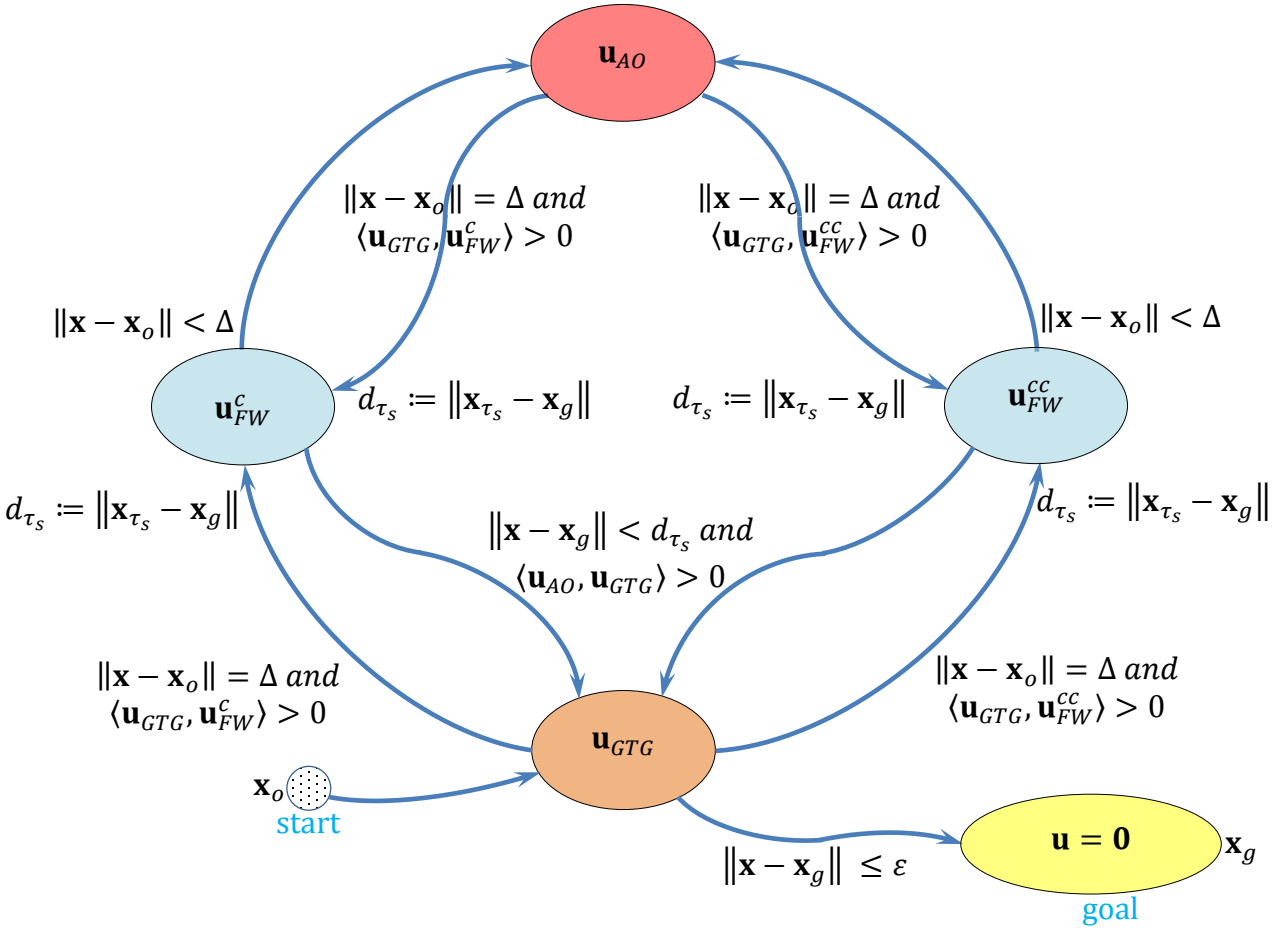


Figure 3.12. Setup for navigation architecture [73].

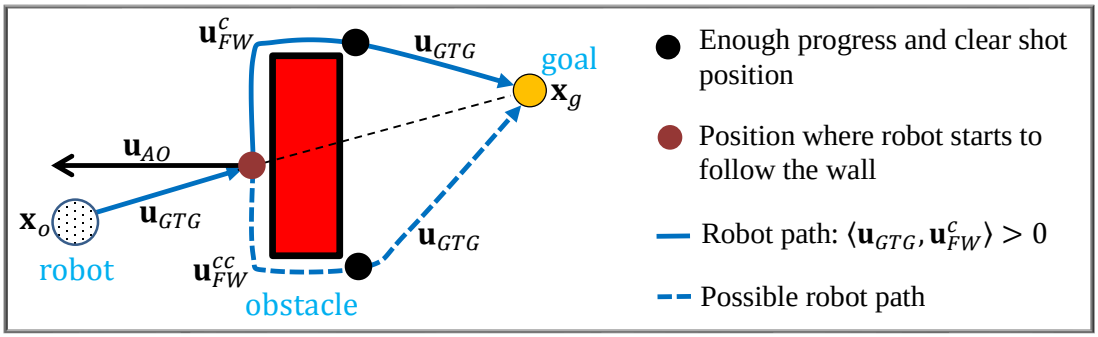


Figure 3.13. Illustration of the navigation system [73].

3.1.2.2.6 Tracking and transformation of the 'simple' model input. The simple planning model input, $\mathbf{u}_m = (u_1\ u_2)^T$ can be tracked using a PID controller or clever transformation can be used to transform it into the unicycle model input, $\mathbf{u} = (v\ \omega)^T$ as shown in Figure 3.14. These two approaches are discussed below.

In the first approach, the tracking is done by using a PID controller. Let the current position of the point mass be $\mathbf{x} = (x\ y)^T$, see Figure 3.8, then

$$\varphi_g = arctan\left(\frac{u_2}{u_1}\right) \tag{3.33}$$

$$\omega = PID\left(\varphi_g - \varphi\right) \tag{3.34}$$

From (3.2)

$$\sqrt{\dot{x}^2 + \dot{y}^2} = \sqrt{v^2 cos^2\varphi + v^2 sin^2\varphi} = v$$

$$\Rightarrow \qquad v = \sqrt{u_1^2 + u_2^2} = \|\mathbf{u}_m\| \tag{3.35}$$

In second method, a clever transformation approach is used where a new point (x_n, y_n), of interest is selected on the robot at a distance k from the center of mass, (x, y), as shown in Figure 3.14, where $\mathbf{x}_n = (x_n\ y_n)^T$ and $\dot{\mathbf{x}}_n = \mathbf{u}_m$. If the orientation is ignored then [73]

$$\begin{aligned} x_n &= x + kcos\varphi \\ y_n &= y + ksin\varphi \end{aligned} \tag{3.36}$$

Thus

$$\begin{aligned} \dot{x}_n &= \dot{x} - k\dot{\varphi}sin\varphi \\ \dot{y}_n &= \dot{y} + k\dot{\varphi}cos\varphi \end{aligned} \tag{3.37}$$

Substituting (3.2) into (3.37), and using $\dot{x}_n = u_1$ and $\dot{y}_n = u_2$, we have

$$\begin{aligned} \dot{x}_n &= vcos\varphi - k\omega sin\varphi = u_1 \\ \dot{y}_n &= vsin\varphi + k\omega cos\varphi = u_2 \end{aligned}$$

$$\Rightarrow \qquad \begin{bmatrix} cos\varphi & -sin\varphi \\ sin\varphi & cos\varphi \end{bmatrix} \begin{bmatrix} v \\ k\omega \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\Rightarrow \qquad R(\varphi)\begin{bmatrix}1 & 0\\ 0 & k\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} = \begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

$$\Rightarrow \qquad \begin{bmatrix}v\\ \omega\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & k^{-1}\end{bmatrix} R(-\varphi)\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{3.38}$$

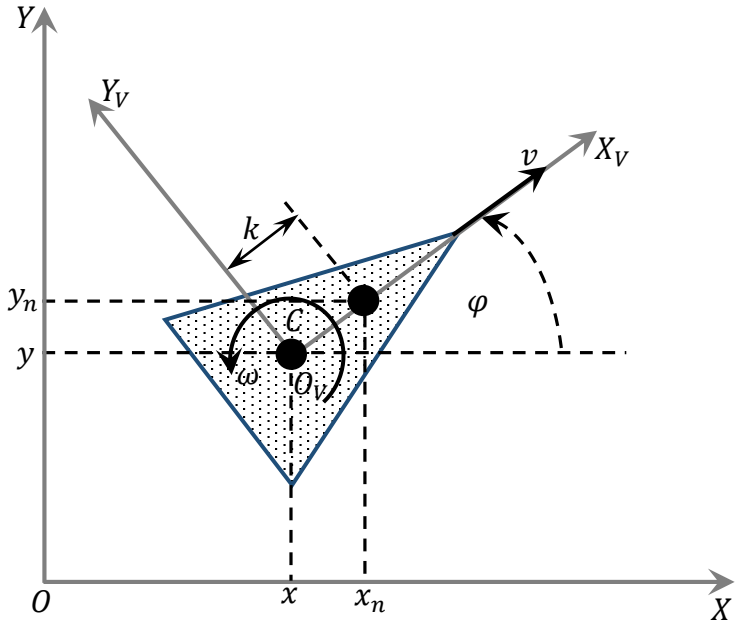


Figure 3.14. Coordinates for DDWMR model for transformation [73].

3.1.3 Challenges. As demonstrated above the non-linear equations of motion can be used to control the WMR, however, some challenges were encountered. The main challenge was in sending signals to the X80SV DC motors and receiving measured data using a developed MATLAB GUI, in real time. Therefore, direction of the research continued to developing autonomous position (x, y, and z) and attitude (roll, pitch, and yaw) control for quadrotor, and then experiment the controllers on the aerial robot, AR.Drone 2.0. The control systems are designed considering the nonlinear effects of control input saturation, time delay, and system uncertainty.

3.2 Aerial Robot: AR.Drone 2.0

3.2.1 Control of quadrotor motions: white-box approach (nonlinear model). This section begins with description of the quadrotor's nonlinear dynamics and kinematics. It is followed by the PD-feedback control laws for the regulation of position and attitude motions.

3.2.1.1 Quadrotor's dynamics and kinematics. The notation for the quadrotor model is shown in Figure 3.15 below. Different coordinate frames are used in identifying quadrotor's location and attitude, and translation and rotation matrices can then be applied to transform one coordinate reference frame into another [74]. Quadrotors has three main coordinate systems attached to it; the body-fixed frame, $\{B\}$, the vehicle frame, $\{V\}$, and the global inertial frame, $\{I\}$. There are two other coordinate systems, not shown, that are of interest, the vehicle-1 frame, $\{V^1\}$, and vehicle-2 frame, $\{V^2\}$ [74]. Frame $\{V^1\}$ is obtained by rotating frame $\{V\}$ about the Z_V-axis by a positive yaw angle, φ_V, assuming no rolling or pitching, so that X_V and Y_V are aligned with X_B and Y_B respectively. Frame $\{V^2\}$ is also obtained by rotating frame $\{V^1\}$ in a right-handed rotation about the Y_{V^1}-axis by the pitch angle, θ_{V^1}, assuming no rolling, so that X_{V^1} and Y_{V^1} are aligned with X_B and Y_B respectively [74].

Unlike most helicopters, quadrotors use two sets of identical fixed pitched propellers; two clockwise and two counter-clockwise (see Figure 3.15). The configuration of the quadrotor is represented by its six degrees of freedom in terms of position, $(x_I, y_I, z_I)^T$, and the attitude defined using the Euler angles, $(\emptyset_{V^2}, \theta_{V^1}, \varphi_V)^T$. This gives a 12-state system characterizing the quadrotor's equations of motion, as $\mathbf{x} = \left(x_I,\ y_I,\ z_I,\ \emptyset_{V^2},\ \theta_{V^1},\ \varphi_V, \dot{x}_B, \dot{y}_B, \dot{z}_B, \dot{\emptyset}_B, \dot{\theta}_B, \dot{\varphi}_B\right)^T$ [39, 74].

The structure of quadrotor in this research is assumed to be rigid and symmetrical, the center of gravity and the body fixed frame origin are coincided, the propellers are rigid and the thrust and drag forces are proportional to the square of propeller's speed [39, 74]. It is also assumed that the earth is flat and non-rotating, which is a valid assumption for quadrotors [39, 74]. The quadrotor has four rotors, labelled 1 to 4, mounted at the end of each cross arm. The rotors are driven by electric motors powered by electronic speed controllers. The vehicle's total mass is m_q and d_q is distance from the motor to the center of mass.

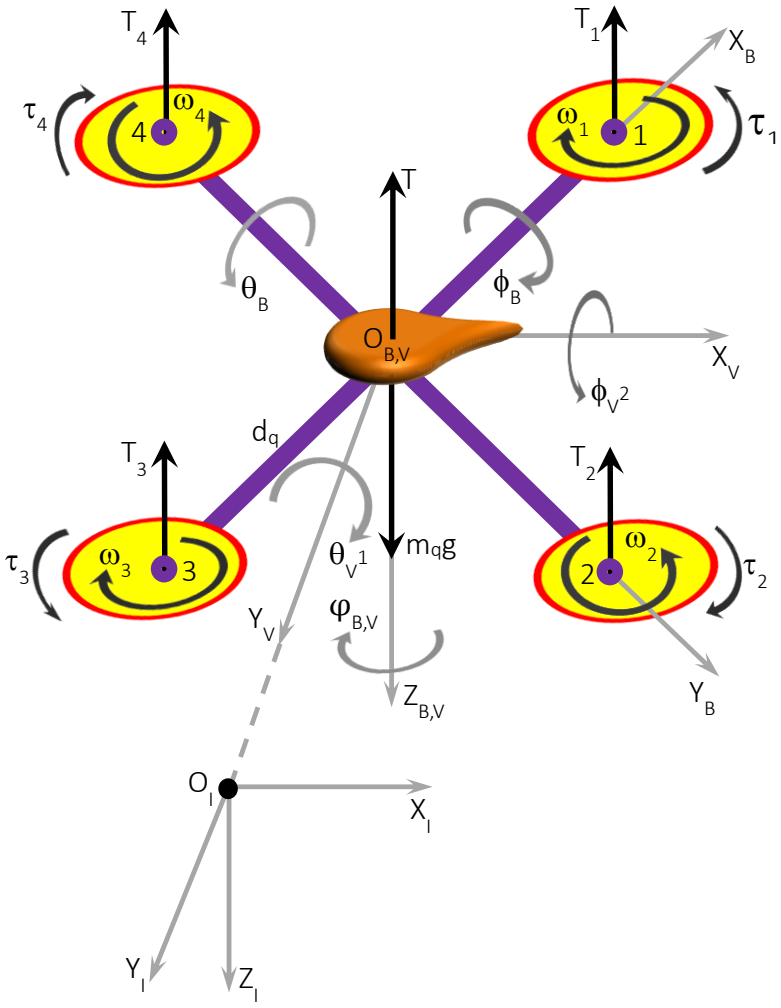


Figure 3.15. Coordinates for the quadrotor.

The forces and the moments on the quadrotor are primarily due to gravity and the four propellers, and since there are no aerodynamic lifting surfaces, it will be assumed that the aerodynamic forces and moments are negligible. The total upward thrust, T, on the vehicle is given by

$$T = \sum_{i=1}^{i=4} T_i \tag{3.39}$$

where the relationship between the rotor speed ω_i and the upward thrust T_i, is defined as [39]:

$$T_i = a\omega_i^2, \quad i = 1, 2, 3, 4 \tag{3.40}$$

where $a > 0$ is the thrust constant that depends on the air density, the cube of the blade radius, the number of blades, and the chord length of the blade [39].

The translational dynamics of the vehicle in the world coordinates is given by Newton's second law [74]

$$m\frac{d\mathbf{v}_B}{dt_I} = \mathbf{F}_B \tag{3.41}$$

where $\mathbf{v}_B = (\dot{x}_B, \dot{y}_B, \dot{z}_B)^T$ is the quadrotor's linear velocity, $\mathbf{F}_B = \left(f_{x_B}, f_{y_B}, f_{z_B}\right)^T$ is the total force applied to the quadrotor, and d/dt_I is the time derivative in frame $\{I\}$. From equation of Coriolis [74], (3.41) becomes

$$m\frac{d\mathbf{v}_B}{dt_I} = m\left(\frac{d\mathbf{v}_B}{dt_B} + \boldsymbol{\omega}_{B/I} \mathrm{x}\, \mathbf{v}_\mathrm{B}\right) = \mathbf{F}_B \tag{3.42}$$

where $\boldsymbol{\omega}_{B/I} = \boldsymbol{\omega}_B = \left(\dot{\phi}_B, \dot{\theta}_B, \dot{\varphi}_B\right)^T$ is the angular velocity of the vehicle in frame $\{B\}$. Now, $\mathbf{F}_B$ is made up of the gravity force, $\mathbf{F}_{g_B}$, and the total thrust from the motors, $\mathbf{F}_{T_B}$, given as

$$\mathbf{F}_B = \mathbf{F}_{g_B} + \mathbf{F}_{T_B} = {}^B\mathbf{R}_V \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -T \end{pmatrix} \tag{3.43}$$

where g is the gravitational acceleration and ${}^B\mathbf{R}_V$ is the rotation matrix from frame $\{V\}$ to frame $\{B\}$ given by (3.57). Substituting (3.43) into (3.42), and rearranging, to obtain

$$\dot{\mathbf{v}}_B = -\boldsymbol{\omega}_B \,\mathrm{x}\, \mathbf{v}_B + {}^B\mathbf{R}_V \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ T/m \end{pmatrix} \tag{3.44}$$

The rotational dynamics of the airframe in frame $\{I\}$ is given by Euler's equation of motion [74]

$$\frac{d\mathbf{h}_B}{dt_I} = \boldsymbol{\Gamma}_B \tag{3.45}$$

where $\mathbf{h}_B = \mathbf{J}\boldsymbol{\omega}_B$ is the quadrotor's angular momentum and $\boldsymbol{\Gamma}_B = (\tau_{x_B}, \tau_{y_B}, \tau_{z_B})^T$ is the applied torque to the airframe. Similarly, using the equation of Coriolis, and rearranging, (3.45) becomes

$$\boldsymbol{\Gamma}_B = \mathbf{J}.\left(\dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \text{ x } \boldsymbol{\omega}_B\right) \tag{3.46}$$

where $\mathbf{J}$ is the constant rotational inertia matrix of the vehicle given by [39, 74]

$$\mathbf{J} = \begin{pmatrix} J_{xx} & -J_{yx} & -J_{zx} \\ -J_{xy} & J_{yy} & -J_{zy} \\ -J_{xz} & -J_{yz} & J_{zz} \end{pmatrix} \tag{3.47}$$

and if the airframe's mass distribution is assumed to be symmetrical with respect to frame $\{B\}$, then $J_{xy} = J_{xz} = J_{yz} = 0$, and $J_{xx} = J_{yy}$. The moments of inertia are calculated as

$$\begin{aligned} J_{xx} = J_{yy} &= \frac{2m_e r^2}{5} + 2d^2 m_c \\ J_{zz} &= \frac{2m_e r^2}{5} + 4d^2 m_c \end{aligned} \tag{3.48}$$

where m_c is mass of the quadrotor's center (assuming a spherical dense center, with radius r) and m_e is the mass at the end of each cross arm, where the propellers are located.

The pairwise differences in rotor thrust cause the vehicle to rotate. The torque about frame $\{B\}$ x-axis, the rolling (positive) torque, is given by

$$\tau_{x_B} = \tau_{\emptyset_B} = d_q(T_4 - T_2) \tag{3.49}$$

Substituting (3.40) into (3.49), to obtain

$$\tau_{\emptyset_B} = a d_q(\omega_4^2 - \omega_2^2) \tag{3.50}$$

and similarly for the y-axis, the pitching (positive) torque is

$$\tau_{y_B} = \tau_{\theta_B} = a d_q(\omega_1^2 - \omega_3^2) \tag{3.51}$$

Due to Newton's third law, the drag of the propellers produces a yawing torque on the body of the quadrotor. The aerodynamic torque is given by [39, 74]

$$\tau_i = \pm b\omega_i^2, \quad i = 1, 2, 3, 4 \tag{3.52}$$

where b is the torque constant, which depends on the same factors as a. This torque exerts a reaction torque on the airframe which acts to rotate the airframe about the propeller shaft in the opposite direction to its rotation. Therefore, the total yawing (positive) torque is given by [33]

$$\tau_{z_B} = \tau_{\varphi_B} = \sum_{i=1}^{i=4} \tau_i = b(\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2) \tag{3.53}$$

where the different signs are due to the different rotation directions of the rotors, thus a yaw torque can be created simply by appropriate coordinated control of all four rotor speeds.

The forces and torques on the quadrotor's airframe, obtained by combining (3.39), (3.50), (3.51), and (3.53), can be written in matrix form as [39]

$$\begin{pmatrix} T \\ \boldsymbol{\Gamma}_B \end{pmatrix} = \mathbf{C} \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} = \mathbf{C}^{-1} \begin{pmatrix} T \\ \tau_{x_B} \\ \tau_{y_B} \\ \tau_{z_B} \end{pmatrix} \tag{3.54}$$

which gives the rotor speeds required to apply a specified thrust T and torque $\boldsymbol{\Gamma}_B$ to the airframe, where

$$\mathbf{C} = \begin{pmatrix} a & a & a & a \\ 0 & -ad_q & 0 & ad_q \\ ad_q & 0 & -ad_q & 0 \\ -b & b & -b & b \end{pmatrix} \tag{3.55}$$

The matrix $\mathbf{C}$ is of full rank if $a, b, d_q > 0$, thus making the vehicle controllable.

The linear velocities in the different frames are related by [74]

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_I = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_V = {}^V\mathbf{R}_B \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_B \tag{3.56}$$

where ${}^V\mathbf{R}_B$ is the transformation matrix from frame $\{B\}$ to frame $\{V\}$ given by [32, 74]

$$^{V}\mathbf{R}_{B} = [^{B}\mathbf{R}_{V}]^{T} = \left[{}^{B}\mathbf{R}_{V^2}(\emptyset)^{V^2}\mathbf{R}_{V^1}(\theta)^{V^1}\mathbf{R}_{V}(\varphi)\right]^{T}$$

$$= \begin{pmatrix} c\theta c\varphi & s\emptyset s\theta c\varphi - c\theta s\varphi & c\emptyset s\theta c\varphi + s\emptyset s\varphi \\ c\theta s\varphi & s\emptyset s\theta s\varphi + c\emptyset c\varphi & c\emptyset s\theta s\varphi - s\emptyset c\varphi \\ -s\theta & s\emptyset c\theta & c\emptyset c\theta \end{pmatrix} \tag{3.57}$$

and the angular velocities are related as [32, 74]

$$\begin{pmatrix} \dot{\emptyset} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix}_{V} = \begin{pmatrix} 1 & s\emptyset tan\theta & c\emptyset tan\theta \\ 0 & c\emptyset & -s\emptyset \\ 0 & -s\emptyset sec\theta & c\emptyset sec\theta \end{pmatrix} \begin{pmatrix} \dot{\emptyset} \\ \dot{\theta} \\ \dot{\varphi} \end{pmatrix}_{B} \tag{3.58}$$

where $s \triangleq sin, c \triangleq cos,\ \emptyset \triangleq \emptyset_{V^2}, \theta \triangleq \theta_{V^1}\ and\ \varphi \triangleq \varphi_{V}$.

3.2.1.2 Control algorithms. Control of the quadrotor input, $(\omega_1, \omega_2, \omega_3, \omega_4)^T$, is about applying the appropriate thrust, T, and/or torque, $\mathbf{\Gamma}_B$, to the airframe, which will be determined using the traditional PID-feedback controller. In general, the dynamics of rotational systems has a second order transfer function of the form [39]

$$\frac{\Theta(s)}{\tau(s)} = \frac{1}{J_r s^2 + B_a s} \tag{3.59}$$

where B_a is the aerodynamic damping, J_r is the rotational inertia, $\Theta(s)$ is the output signal (e.g. pitch), and $\tau(s)$ is the input signal (e.g. pitching torque). B_a is generally quite small, thus the integral controller is not needed in the regulation of the quadrotor, and therefore PD controller is applied in this section of the research.

3.2.1.2.1 Attitude controllers. This section presents control algorithms that make quadrotor pitch, roll, and yaw. The PD controller is used to determine the required torques based on the error between desired angles ($\theta_B^*, \emptyset_B^*$, and φ_B^*) and actual angles ($\theta_B, \emptyset_B$, and φ_B). For real-time application, the actual vehicle angles would be estimated by an inertia navigation system. The required rotor speeds are then calculated from the respective torques using (3.54). Typically, the rate of the desired angles ($\dot{\theta}_B^*$, $\dot{\emptyset}_B^*$, and $\dot{\varphi}_B^*$) are small, and can be ignored.

The pitch motion is controlled by increasing the speed, which in turn increases the thrust, of either rotor 1 or 3, while keeping the speed of rotor 2 or 4 the same or zero, as illustrated in Figure 3.16. The required pitching torque on the airframe is given by

$$\tau_{y_B} = \tau_{\theta_B} = K_{P_\theta}(\theta_B^* - \theta_B) + K_{D_\theta}\left(\dot{\theta}_B^* - \dot{\theta}_B\right)$$

$$\Rightarrow \qquad \tau_{\theta_B} = K_{P_\theta}\left[(\theta_B^* - \theta_B) - K_{D_\theta}/K_{P_\theta}\left(\dot{\theta}_B\right)\right] \tag{3.60}$$

The roll motion is controlled by increasing the speed, which in turn increases the thrust of either rotor 2 or 4, while keeping the speed of rotor 1 or 3 the same or zero, as illustrated in Figure 3.17. The required rolling torque on the airframe is given by

$$\tau_{x_B} = \tau_{\emptyset_B} = K_{P_\emptyset}(\emptyset_B^* - \emptyset_B) + K_{D_\emptyset}\left(\dot{\emptyset}_B^* - \dot{\emptyset}_B\right)$$

$$\Rightarrow \qquad \tau_{\emptyset_B} = K_{P_\emptyset}\left[(\emptyset_B^* - \emptyset_B) - K_{D_\emptyset}/K_{P_\emptyset}\left(\dot{\emptyset}_B\right)\right] \tag{3.61}$$

The yaw motion is controlled by simultaneously applying the same speed, which in turn changes the thrusts of rotors 2 and 4 while keeping speeds of rotors 1 and 3 the same or zero, or vice versa, as illustrated in Figure 3.18. The required yawing torque on the airframe is given by

$$\tau_{z_B} = \tau_{\varphi_B} = K_{P_\varphi}(\varphi_B^* - \varphi_B) + K_{D_\varphi}(\dot{\varphi}_B^* - \dot{\varphi}_B)$$

$$\Rightarrow \qquad \tau_{\varphi_B} = K_{P_\varphi}\left[(\varphi_B^* - \varphi_B) - K_{D_\varphi}/K_{P_\varphi}\left(\dot{\varphi}_B\right)\right] \tag{3.62}$$

3.2.1.2.2 Position controllers. This section presents control algorithms that make quadrotor undergoes x, y, and z motions. Again, PD controller is used to determine the required control inputs based on the error between desired positions (x_I^*, y_I^*, and z_I^*) and actual positions (x_I, y_I, and z_I). For real-time application, the actual vehicle positions and velocities would be estimated by an inertia navigation systems or GPS receivers. The required rotor speeds are then calculated from the respective torques using (3.54). Once again, rate of the desired positions ($\dot{x}_I^*$, $\dot{y}_I^*$, and $\dot{z}_I^*$) are small, and can be ignored.

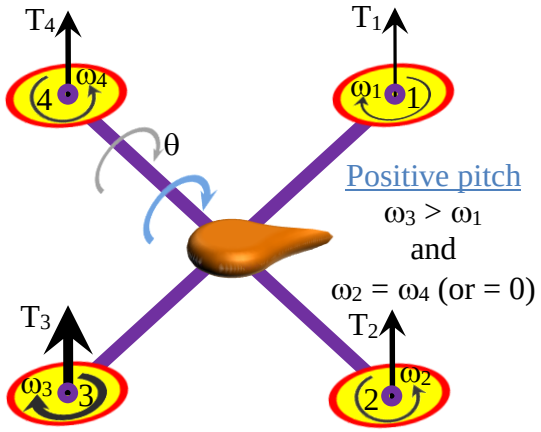


Figure 3.16. Coordinates for pitch motion.

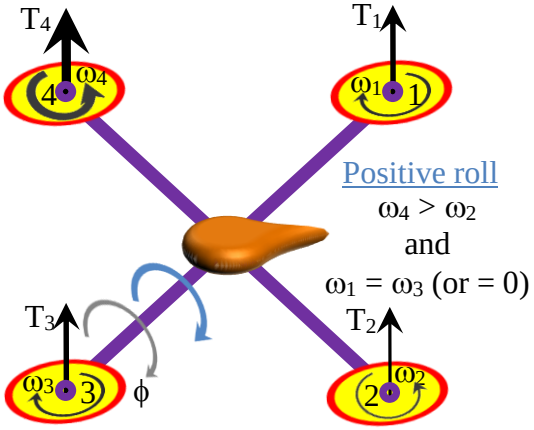


Figure 3.17. Coordinates for roll motion.

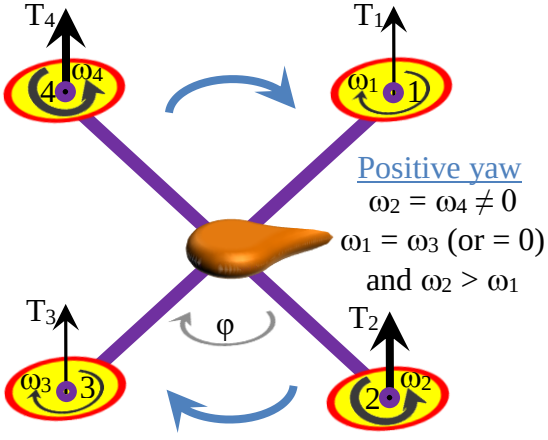


Figure 3.18. Coordinates for yaw motion.

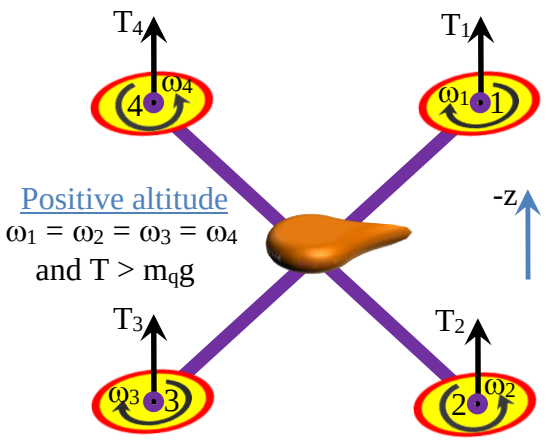


Figure 3.19. Coordinates for altitude motion.

The altitude motion is controlled by simultaneously applying the same speed, which in turn changes the thrusts, of all the four rotors, as illustrated in Figure 3.19. For upward motion, T must be bigger than the weight, $m_q g$, of the quadrotor, taken into account the drag on the vehicle. The total thrust on the airframe is given by

$$T = K_{P_z}(z_I^* - z_I) + K_{D_z}(\dot{z}_I^* - \dot{z}_I) + T_o$$

$$\Rightarrow \qquad T = K_{P_z}\left[(z_I^* - z_I) - K_{D_z}/K_{p_z}\,(\dot{z}_I)\right] + T_o \tag{3.63}$$

where the additive term is given as

$$T_o = m_q g = 4a\omega_o^2 \tag{3.64}$$

and ω_o is the average rotor speed necessary to generate a thrust, T_o, equal to the weight of the vehicle; a feed-forward control approach – used to counter the effect of gravity, which otherwise is a constant disturbance to the altitude motion. The alternative approach would be to have very high gains for the PD controller. This second approach, not used in this research, often leads to actuator saturation and instability [39].

To move the vehicle in the x-direction (along X_V) its nose pitch down, which generates a force [39]

$$\boldsymbol{f} = R_{y_V}(\theta_V)\begin{pmatrix}0\\0\\T\end{pmatrix} = \begin{pmatrix}Tsin\theta_V\\0\\Tcos\theta_V\end{pmatrix} \tag{3.65}$$

which gives the force component that accelerates the vehicle in the X_V-direction as

$$f_{x_V} = Tsin\theta_V \approx T\theta_V \tag{3.66}$$

where θ_V is small. The velocity in this direction can be controlled using the P controller [39]

$$f_{x_v}^* = m_q K_{f_x}\left(v_{x_V}^* - v_{x_V}\right) \tag{3.67}$$

Combining (3.66) and (3.67), the desired pitch angle required to achieve the desired forward velocity is obtained as

$$\theta_V^* = \frac{m_q K_{f_x}}{T}\left(v_{x_V}^* - v_{x_V}\right) \tag{3.68}$$

For a vehicle in vertical equilibrium the total thrust is equal to the weight of the airframe, thus $m_q/T = m_q/T_o$ in (3.68) can be approximately substituted by g^{-1}. Now, the desired velocity in frame $\{V\}$ relative to frame $\{I\}$ is then determined as [39]

$$v_{x_V}^* = {}^V\boldsymbol{R}_I(\varphi)v_{x_I}^* = [\,{}^I\boldsymbol{R}_V]^T(\varphi)v_{x_I}^* \tag{3.69}$$

where the desired velocity in frame $\{I\}$ is given by the P controller

$$v_{x_I}^* = K_{P_{x_I}}(x_I^* - x_I) \tag{3.70}$$

Thus, to reach a desired x-position, the appropriate velocity is calculated and from that the appropriate pitch angle which will generate the force to move the vehicle is obtained. This indirection is consequence of the vehicle being under-actuated – the vehicle has just four rotor speeds to adjust but its configuration space is 6-dimensional. Substituting (3.70) into (3.69), and then the result into (3.68), a compact form control algorithm for computing the desired pitch angle can be obtained as

$$\theta_V^* = K_{P_x}\left([\,{}^I\boldsymbol{R}_V]^T(\varphi)(x_I^* - x_I) - K_{D_x}\dot{x}_V\right) \tag{3.71}$$

Similar analysis can be carried out to obtain the control algorithm to move the vehicle in the y-direction (along Y_V). To reach the desired y-position, the appropriate velocity, $v_{y_V}^*$, is calculated, and from that the appropriate roll angle, $\emptyset_V^*$, which will generate the force to move the vehicle is obtained. The compact form control algorithm for computing the desired roll angle is also given as

$$\emptyset_V^* = K_{P_y}\left([\ {}^I\boldsymbol{R}_V]^T(\varphi)(y_I^* - y_I) - K_{D_y}\dot{y}_V\right) \tag{3.72}$$

To move the vehicle forward or sideway its airframe must first pitch down or roll down respectively so that the thrust vector has a horizontal component to accelerate it; the total thrust must be increased so that the vertical thrust component still balances gravity. As it approaches its goal, the airframe must be rotated in the opposite direction, pitching up or rolling up, so that the backward component of the thrust decelerates the forward or the sideway motion. Finally, the airframe rotates to the horizontal with the thrust vector vertical. The cost of the under-actuation is once again manoeuvre. The pitch and the roll angles cannot be arbitrarily set, they are means to achieve the translation control. (3.71) and (3.72) can be combined as

$$\begin{pmatrix}\theta_V^*\\ \emptyset_V^*\end{pmatrix} = \begin{pmatrix}K_{P_x}\\ K_{P_y}\end{pmatrix} * \left([\ {}^I\boldsymbol{R}_V]^T(\varphi)(\boldsymbol{p}_I^* - \boldsymbol{p}_I) - K_{D_{xy}}\begin{pmatrix}\dot{x}_V\\ \dot{y}_V\end{pmatrix}\right) \tag{3.73}$$

where $\boldsymbol{p}_I = (x_I, y_I)^T$, $K_{D_x} = K_{D_y} = K_{D_{xy}}$, and

$$[\ {}^I\boldsymbol{R}_V]^T(\varphi) = [\ {}^I\boldsymbol{R}_V]^{-1}(\varphi) = \begin{pmatrix}c\varphi & -s\varphi\\ s\varphi & c\varphi\end{pmatrix} \tag{3.74}$$

3.2.2 Challenges. As demonstrated above, the nonlinear equations of motion can be used to control the position of the quadrotor. However, they are not appropriate for the control design for obvious reason that they are too complicated to gain significant insight into the motions. Moreover, the approach above used MATLAB *S-function* to implement the dynamics and kinematics of the quadrotors, and this approach resulted in some difficulties in terms of C/C^{++} code

generation for real-time application. Therefore, the research focus on obtaining a linearized model for the quadrotor's position, and for this research the altitude motion was considered.

3.2.3 Control of quadrotor altitude motion: white-box approach (linearized model). This section begins with obtaining altitude linearized model from the quadrotor's dynamics and kinematics. It is followed by determining state-space representation and the transfer function for the plant model. Then checks for the stability, controllability and observability are carried out, as well as determination of the equilibrium points of the system. The last part deals with specifying design criteria for the altitude response, and presenting the various controllers that have been designed.

3.2.3.1 Quadrotor altitude linearized model. Now, taking upward as the positive direction for the z-axis (altitude), equation (3.44) can be written as [74]

$$\begin{pmatrix}\dot{x}_B\\ \dot{y}_B\\ \dot{z}_B\end{pmatrix}=\begin{pmatrix}\dot{\varphi}_B\dot{y}_B-\dot{\theta}_B\dot{z}_B\\ \dot{\emptyset}_B\dot{z}_B-\dot{\varphi}_B\dot{x}_B\\ \dot{\theta}_B\dot{x}_B-\dot{\emptyset}_B\dot{y}_B\end{pmatrix}+\begin{pmatrix}0\\ 0\\ T/m_q\end{pmatrix}-{}^{B}\mathbf{R}_V\begin{pmatrix}0\\ 0\\ g\end{pmatrix} \tag{3.75}$$

Substituting (3.57) into (3.75), to obtain

$$\begin{pmatrix}\ddot{x}_B\\ \ddot{y}_B\\ \ddot{z}_B\end{pmatrix}=\begin{pmatrix}\dot{\varphi}_B\dot{y}_B-\dot{\theta}_B\dot{z}_B\\ \dot{\emptyset}_B\dot{z}_B-\dot{\varphi}_B\dot{x}_B\\ \dot{\theta}_B\dot{x}_B-\dot{\emptyset}_B\dot{y}_B\end{pmatrix}+\begin{pmatrix}0\\ 0\\ T/m_q\end{pmatrix}-\begin{pmatrix}-gs\theta\\ gc\theta s\emptyset\\ gc\theta c\emptyset\end{pmatrix} \tag{3.76}$$

and substituting (3.57) into (3.56), to obtain

$$\begin{pmatrix}\dot{x}_I\\ \dot{y}_I\\ \dot{z}_I\end{pmatrix}=\begin{pmatrix}c\theta c\varphi & s\emptyset s\theta c\varphi-c\theta s\varphi & c\emptyset s\theta c\varphi+s\emptyset s\varphi\\ c\theta s\varphi & s\emptyset s\theta s\varphi+c\emptyset c\varphi & c\emptyset s\theta s\varphi-s\emptyset c\varphi\\ -s\theta & s\emptyset c\theta & c\emptyset c\theta\end{pmatrix}\begin{pmatrix}\dot{x}_B\\ \dot{y}_B\\ \dot{z}_B\end{pmatrix} \tag{3.77}$$

Differentiating (3.77) and neglecting ${}^{V}\dot{\mathbf{R}}_B$, and also neglecting the Coriolis terms in (3.76), to obtain

$$\begin{pmatrix}\ddot{x}_I\\ \ddot{y}_I\\ \ddot{z}_I\end{pmatrix}=\begin{pmatrix}c\theta c\varphi & s\emptyset s\theta c\varphi-c\theta s\varphi & c\emptyset s\theta c\varphi+s\emptyset s\varphi\\ c\theta s\varphi & s\emptyset s\theta s\varphi+c\emptyset c\varphi & c\emptyset s\theta s\varphi-s\emptyset c\varphi\\ -s\theta & s\emptyset c\theta & c\emptyset c\theta\end{pmatrix}\begin{pmatrix}\ddot{x}_B\\ \ddot{y}_B\\ \ddot{z}_B\end{pmatrix} \tag{3.78}$$

$$\begin{pmatrix} \ddot{x}_B \\ \ddot{y}_B \\ \ddot{z}_B \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ T/m_q \end{pmatrix} - \begin{pmatrix} -gs\theta \\ gc\theta s\emptyset \\ gc\theta c\emptyset \end{pmatrix} \tag{3.79}$$

Substituting (3.79) into (3.78), to obtain

$$\begin{pmatrix} \ddot{x}_I \\ \ddot{y}_I \\ \ddot{z}_I \end{pmatrix} = \begin{pmatrix} c\theta c\varphi & s\emptyset s\theta c\varphi - c\theta s\varphi & c\emptyset s\theta c\varphi + s\emptyset s\varphi \\ c\theta s\varphi & s\emptyset s\theta s\varphi + c\emptyset c\varphi & c\emptyset s\theta s\varphi - s\emptyset c\varphi \\ -s\theta & s\emptyset c\theta & c\emptyset c\theta \end{pmatrix} \left[\begin{pmatrix} 0 \\ 0 \\ T/m_q \end{pmatrix} - \begin{pmatrix} -gs\theta \\ gc\theta s\emptyset \\ gc\theta c\emptyset \end{pmatrix} \right] \tag{3.80}$$

and for altitude motion no pitching and rolling, and assuming the quadrotor can yaw, (3.80) becomes

$$\begin{pmatrix} \ddot{x}_I \\ \ddot{y}_I \\ \ddot{z}_I \end{pmatrix} = \begin{pmatrix} c\varphi & -s\varphi & 0 \\ s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \left[\begin{pmatrix} 0 \\ 0 \\ T/m_q \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \right]$$

$$\Rightarrow \qquad \begin{pmatrix} \ddot{x}_I \\ \ddot{y}_I \\ \ddot{z}_I \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ T/m_q - g \end{pmatrix} \tag{3.81}$$

and taking the equation of motion only in the z-direction, to obtain

$$\ddot{z}(t) = \frac{T(t)}{m_q} - g \tag{3.82}$$

Thus, to control the altitude, $z(t)$, of the quadrotor only the total thrust, $T(t)$, need to be varied, since m_q and g are constants. The altitude is controlled by simultaneously applying the same speed, which in turn changes the thrusts of all the four rotors, as illustrated in Figure 3.19 above.

Now, if the quadrotor is in vertical equilibrium, that is when the quadrotor is at a reference point, z_o, on the ground or hover position, the propellers generate a total thrust, T_o, equal to the weight of the vehicle as shown in (3.64), and if the average rotor speed necessary to generate a total thrust is $\omega(t)$, then from (3.40)

$$T(t) = 4\mathrm{a}\omega^2(\mathrm{t}) \tag{3.83}$$

Substituting (3.83) into (3.82), we obtain

$$\ddot{z}(t) = \frac{4a\omega^2(t)}{m_q} - g \tag{3.84}$$

which is a nonlinear dynamic system because of the term g.

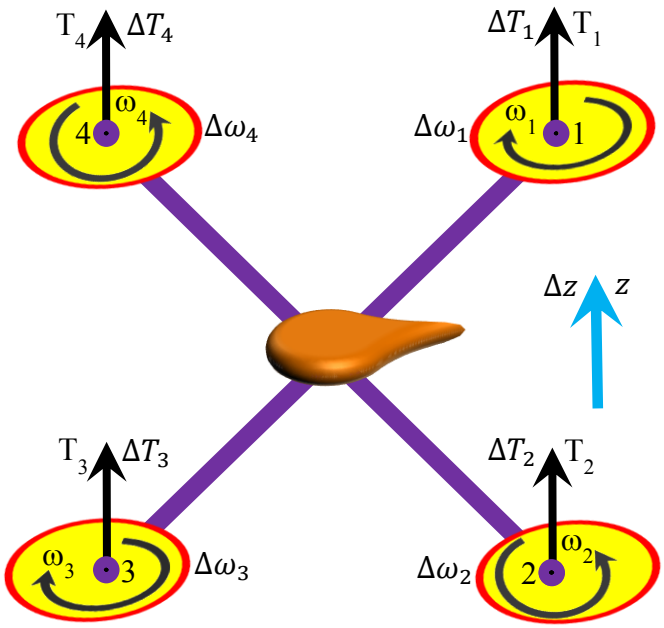


Figure 3.20. Coordinates for altitude motion: white-box approach (linearized model).

To linearize the system about any equilibrium point where the quadrotor will hover, which will be a desired height, z_o, consider a change in the average rotor speed, $\Delta\omega(t)$, that will produce a change in the total thrust, $\Delta T(t)$, needed to cause the quadrotor to rise or drop by $\Delta z(t)$, see Figure 3.20, then (3.84) becomes

$$\ddot{z_o} + \ddot{\Delta z}(t) = \frac{4a\left(\omega_o + \Delta\omega(t)\right)^2}{m_q} - g \tag{3.85}$$

Expanding and neglecting the term $(\Delta\omega(t))^2$, (3.85) becomes

$$\ddot{\Delta z}(t) \cong \left(\frac{4a\omega_o^2 + 8a\omega_o\Delta\omega(t)}{m_q}\right) - g \tag{3.86}$$

Combining (3.64) and (3.83), and substituting into (3.86), and then simplifying, to obtain

$$\ddot{\Delta z}(t) \cong \frac{8a\omega_o \Delta\omega(t)}{m_q}$$

$$\Rightarrow \qquad \ddot{\Delta z}(t) \cong \Delta\omega(t)\sqrt{\frac{16ga}{m_q}} \tag{3.87}$$

which gives the relationship between change in the average rotor and the vertical acceleration of the quadrotor. It is assumed that $\Delta\omega(t)$ is not too large because the model would eventually be influenced by strong nonlinearities and saturations.

3.2.3.2 State-space and transfer function models. The altitude linear model shown in (3.87) is a SISO continuous LTI control system with a control signal input, $u(t) = \Delta\omega(t)$, and the measured output signal for feedback is $y(t) = \Delta z(t)$. Let the states of the system be $x_1(t)$ and $x_2(t)$ equals $\Delta z(t)$ and $\dot{\Delta z}(t)$, respectively, then

$$\begin{aligned} \dot{x}_1(t) &= \dot{\Delta z}(t) = x_2(t) \\ \dot{x}_2(t) &= \ddot{\Delta z}(t) = u(t)\sqrt{16ga/m_q} \end{aligned} \tag{3.88}$$

which can be written in matrix form as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ \sqrt{16ga/m_q} \end{bmatrix} u(t) \\ y(t) &= [1 \quad 0]\mathbf{x}(t) \end{aligned} \tag{3.89}$$

where $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 4\sqrt{ga/m_q} \end{bmatrix}, \mathbf{C} = [1 \quad 0], \mathbf{D} = [0]$, and $\mathbf{x}(t) = [x_1(t) \;\; x_2(t)]^T$. The transfer function of the system was obtained using the MATLAB command $ss2tf(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, as

$$G_p(s) = \frac{Y(s)}{U(s)} = \frac{4}{s^2}\sqrt{\frac{ga}{m_q}} \tag{3.90}$$

3.2.3.3 Stability, equilibrium points, controllability, and observability. The section checks for the stability, equilibrium points, controllability, and observability of the plant model.

3.2.3.3.1 Stability. The poles of the system in (3.90) are s = 0 (repeated), thus the system is unstable, which is also confirmed by the plant's step response shown in Figure 3.21 below.

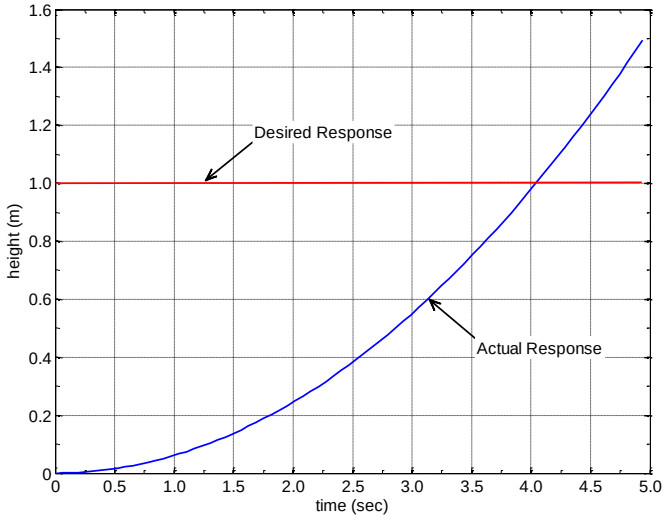


Figure 3.21. Open-loop plant step response: white-box approach (linearized model).

3.2.3.3.2 Equilibrium points. A state $\mathbf{x}_e$ is an equilibrium when $\mathbf{x}(t) = \mathbf{x}_e$, and it remains equal for all future times, and this is obtained when $\dot{\mathbf{x}}(t) = \mathbf{0}$, from the autonomous system, $\mathbf{A}\mathbf{x}_e(t) = 0$ in (3.89). The LTI system has infinity of equilibrium points, given as

$$\mathbf{x}_e = \begin{pmatrix} \Delta z_{ref} \\ 0 \end{pmatrix} \tag{3.91}$$

where Δz_{ref} is infinite quadrotor desired heights, and these equilibria can only be achieved when the linear vertical velocity at these points are zero.

3.2.3.3.3 Controllability and observability. The controllability matrix, $CONT$, of the system was determined using the MATLAB command $ctrb(\mathbf{A}, \mathbf{B})$, given as

$$CONT = \begin{bmatrix} 0 & 4\sqrt{ga/m_q} \\ 4\sqrt{ga/m_q} & 0 \end{bmatrix} \tag{3.92}$$

and observability matrix, $OBSER$, of the system was determined using the MATLAB command $obsv(\mathbf{A}, \mathbf{C})$, given as

$$OBSER = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.93}$$

The ranks of both $CONT$ and $OBSER$ is 2, which is the dimension of the state matrix, **A**, therefore the system is completely state controllable and completely observable, and thus the system is stabilizable and detectable.

3.2.3.4 Performance specifications. The performance specifications for this system will be based on time-domain specifications of the transient response as well as a steady-state requirement, by applying a desired altitude of $1.0m$ step-input for under 7 seconds of simulation time. The AR.Drone 2.0 flight management system sampling time, T_s, is $0.065s$, which is the fixed time-step at which the control law is executed and the navigation data received. The design criteria are as follows: $t_s < 3.0s$ (2% error band), $t_p < \pi s$, $M_o < 3.0\%$, and $|E_{ss}| < 1\%$.

3.2.3.5 Design of controllers. This section presents the design and control system setup for the various controllers used for the simulations and experiments with and without nonlinear effects of saturation, time delay, and system uncertainty.

3.2.3.5.1 Pole placement controller without nonlinear effects. Initially, MATLAB $sisotool$ command was used to determine the region in the left-half s-plane where the desired closed-loop poles (DCLP), **I**, are to be selected to satisfy the transient response specifications, see Figure 3.22. The **I** values were selected within this region, and then MATLAB $acker(\mathbf{A}, \mathbf{B}, \mathbf{I})$ command was used to determine **K** values, for a given **I**, and for the given system parameters **A** and **B**. The MATLAB *acker* function is a pole placement feedback gain matrix calculator that uses the *Ackermann's* formula, and it is only applicable to SISO systems. Finally, the calculated **K** values are used to examine a unit-step response of the closed-loop system, to check if all the performance specifications are satisfied.

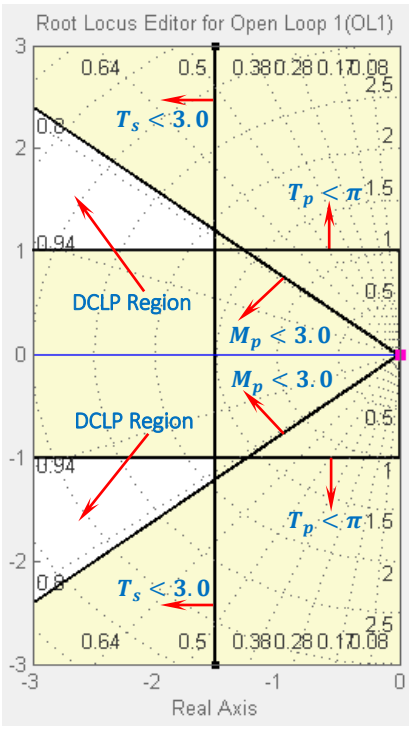


Figure 3.22. s-complex plane showing desired close-loop poles (DCLP) region.

3.2.3.5.2 LQR controller without nonlinear effects. The weight on the control effort was selected as $R = 1$, since the control input, $u(t)$, was initially unconstrained, and different ways were used to select the weight on the error, given as

$$\mathbf{Q} = \mu \mathbf{C}^T \mathbf{C} = \begin{bmatrix} \mu & 0 \\ 0 & 0 \end{bmatrix}, \tag{3.94}$$

$$\mathbf{Q} = \begin{bmatrix} \mu & 0 \\ 0 & 1 \end{bmatrix}, \tag{3.95}$$

$$\mathbf{Q} = \begin{bmatrix} 2\mu & 0 \\ 0 & \mu \end{bmatrix}, \tag{3.96}$$

$$and \qquad \mathbf{Q} = \begin{bmatrix} n_v\mu & 0 \\ 0 & \mu \end{bmatrix}, \tag{3.97}$$

where $\mu \geq 1$ is a weighting constant and n_v a constant to be varied. Different values of μ and n_v were selected, and then MATLAB $lqr(\mathbf{A}, \mathbf{B}, \mathbf{Q}, \mathbf{R})$ command was used to determine the optimal $\mathbf{K}$. The MATLAB *lqr* function is an optimal feedback gain matrix calculator that solves the algebraic

Riccati equation by minimizing a cost function that depends on weights **Q**, **R**, and **N**, given system parameters **A** and **B**. Finally, the calculated **K** is used to examine a unit-step response of the closed-loop system, to check if all the performance specifications are satisfied.

3.2.3.5.3 MIT rule MRAC controller without nonlinear effects. The reference model is to be selected based on the transient specifications, given in the form as

$$G_m(s) = \frac{a_{om}}{s^2 + a_{1m}s + a_{om}} \tag{3.98}$$

The control input is given as

$$U = \theta_1 R - \theta_2 y_{plant} \tag{3.99}$$

Substituting $y_{plant} = G_p U$ into (3.99), and simplifying, we obtain

$$U = \frac{\theta_1 R}{1 + \theta_2 G_p} \tag{3.100}$$

and substituting (3.100) into (2.11), to obtain

$$e(\theta) = \frac{G_p \theta_1 R}{1 + \theta_2 G_p} - G_m R \tag{3.101}$$

Differentiating (3.101) with respect θ_1 and θ_2, to obtain

$$\frac{\partial e}{\partial \theta_1} = \left(\frac{G_p}{1 + \theta_2 G_p}\right) R \tag{3.102a}$$

$$\frac{\partial e}{\partial \theta_2} = -\left(\frac{G_p}{1 + \theta_2 G_p}\right)\left(\frac{G_p \theta_1 R}{1 + \theta_2 G_p}\right) = -\left(\frac{G_p}{1 + \theta_2 G_p}\right) y_{plant} \tag{3.102b}$$

Substituting G_p from (3.90) into (3.102), to obtain

$$\frac{\partial e}{\partial \theta_1} = \left(\frac{4\sqrt{ga/m_q}}{s^2 + 4\theta_2\sqrt{ga/m_q}}\right) R \tag{3.103a}$$

$$\frac{\partial e}{\partial \theta_2} = -\left(\frac{4\sqrt{ga/m_q}}{s^2 + 4\theta_2\sqrt{ga/m_q}}\right) y_{plant} \tag{3.103b}$$

If the reference model is close to the plant, then their characteristic equations can be equally approximated as

$$s^2 + 4\theta_2\sqrt{ga/m_q} \approx s^2 + a_{1m}s + a_{om} \tag{3.104}$$

then MIT update rule, from (2.13), (3.103), and (3.104), can be written as

$$\frac{d\theta_1}{dt} = -\gamma\left(\frac{a_{1m}s + a_{om}}{s^2 + a_{1m}s + a_{om}}R\right)e \tag{3.105a}$$

$$\frac{d\theta_2}{dt} = \gamma\left(\frac{a_{1m}s + a_{om}}{s^2 + a_{1m}s + a_{om}}y_{plant}\right)e \tag{3.105b}$$

3.2.3.5.4 Controllers with saturation effects. The angular velocity of each of the four rotors of the drone is up to $\pm 250 rads^{-1}$, which produce a rotor thrust up to about $\pm 3N$ with a possible total thrust of about $\pm 12N$. Also, for experiments and from the AR.Drone 2.0 software development kit (SDK) documentation, the control input is the vertical speed, $\dot{\Delta z}(t)$, has to be constrained to $[-1\ \ 1]\ ms^{-1}$, to prevent damage.

The dynamics in the z-direction, as shown in (3.87), is dependent on the average change in rotor speeds, $\Delta\omega(t)$, of all the rotors. By inspection of the control input and the vertical speed plots from the pole placement controller, without constraints, $\Delta\omega(t) = \pm 50 rads^{-1}$ seems appropriate to achieve $\dot{\Delta z}(t) = \pm 1 ms^{-1}$. The control input constraints were achieved by inserting a *saturation block* into the Simulink model. To further check whether this is appropriate constraints for $\Delta\omega(t)$, different constraints were applied to the pole placement close-loop system, with values taken at $\pm 10\ rads^{-1}$, $\pm 20\ rads^{-1}$, $\pm 30\ rads^{-1}$, and $\pm 50\ rads^{-1}$. The effects on the altitude and vertical speed plots were also inspected, and a suitable saturation values of $\Delta\omega(t) = \pm 50\ rads^{-1}$ were applied in designing the rest of the control systems.

3.2.3.5.5 Controllers with time delay effects. The overall time delay, T_d, in the control system due to communication between the drone and host computer was implemented as an

actuator time delay (see Figure 2.14) using Simulink *transport delay block*, with only parameters settings being the time delay value and the initial buffer size. The initial buffer size defines the initial memory allocation for the number of input points to store, and if the number of input points exceeds this value, the block allocates additional memory. The default buffer size of 1024 will be used for this research work. The investigation of the time delay effects was carried out for the pole placement and MIT rule MRAC controllers by varying T_d, and checking the behavior of the altitude response. The time delay in the drone's control system is attributed to the:

- processing capability of the host computer,
- electronic devices processing the motion signals,
- measurement reading devices, e.g., the distance between the ultrasonic sensor, for reading the altitude, and the surface can affect the delay,
- processing capability of the control program, e.g., the experiments on the UAV were conducted using MATLAB/Simulink, and the navigation data (yaw, pitch, altitude, battery level, etc.) decoding process contributes to the delay. Also, the different types of solvers introduces delay.

3.2.4 Time-delay estimation: first-order model. Here, we dealing with a continuous-time and an automatic control feedback system so we have an active TDE problem, thus the time delay to be estimated is an explicit parameter in the model. Also, the time delay is not restricted to be a multiple of the sampling interval, but can be a subsample time delay. Therefore, the continuous-time one-step explicit, *idproc*, method will be used in estimating the time delay.

The setup used to control the drone's altitude motion using MATLAB/Simulink program is shown in Figure 3.23. The error between the desired reference input, $z_{des}(t)$, and the system altitude response, $z(t)$, is denoted as $e(t)$. The altitude motion dynamics is implemented using (3.84), used to determine $\omega(t)$ from $\dot{z}(t)$, which is obtained from $\dot{z}_{ref}(t)$, a reference vertical

speed to the drone. The rotors rotate with the same $\omega(t)$, which generate $T(t)$ to produce $z(t)$. These computations takes place on board the drone control engine program written in C.

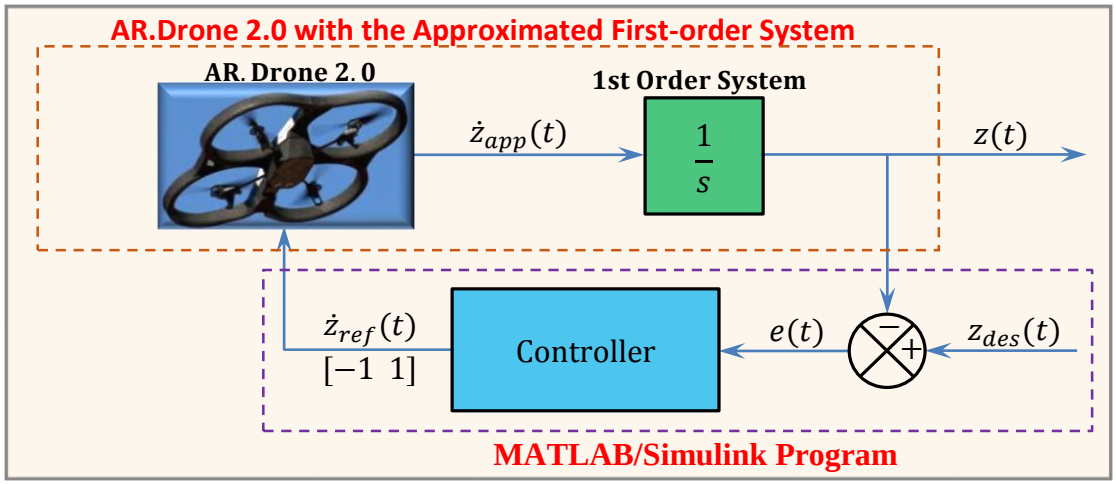


Figure 3.23. AR.Drone 2.0 altitude control system.

The drone's DC motors dynamics are assumed to be very fast such that the altitude control system can be represented as a first-order system using an integrator (see Figure 3.23). Under such assumption, the control input, $\dot{z}_{app}(t)$, to the first-order system is approximately equal to $\dot{z}_{ref}(t)$. Thus, a first-order model is used for the analytical determination of the time delay and for obtaining the simulation altitude responses.

P-feedback controller is used to generate vertical speed signal to control the system's altitude in the determination of the time delay. The transfer function of the time-delayed closed-loop system (see Figure 5.19) for the P controller is given by

$$\frac{Z(s)}{Z_{des}(s)} = \frac{K_P e^{-sT_d}}{s + K_P e^{-sT_d}} \tag{3.106}$$

This time-delay system is a retarded type. As expected, the characteristic equation is transcendental, and therefore the closed-loop poles are infinite; the exponential term in the characteristic equation will introduce oscillations into the system.

3.2.4.1 Analytical method. Consider the first-order scalar homogenous DDE shown in (3.107), see (2.19) for retarded-type delay system. Unlike Ordinary Differential Equations (ODEs), two initial conditions need to be specified for DDEs: a preshape function, $g(t)$, for $-T_d \leq t < 0$, and initial point, z_o, at time, $t = 0$.

$$\dot{z}(t) - b_o z(t) - b_1 z(t - T_d) = 0 \tag{3.107}$$

The characteristic equation of (3.107) is given by

$$s - b_o - b_1 e^{-sT_d} = 0 \tag{3.108}$$

The exponential term, e^{-sT_d}, induced by the time delay term $z(t - T_d)$, makes the characteristic equation transcendental (i.e., infinite dimensional and nonlinear). Thus, it is not feasible to find roots of (3.108), which has an infinite number of roots. Using the Lambert W function defined in equation (2.22), the characteristic equation in (3.108) is solved as [58]

$$s = \frac{1}{T_d} W(T_d b_1 e^{-b_o T_d}) + b_o \tag{3.109}$$

As seen in (3.109), the characteristic root, s, is obtained analytically in terms of parameters, b_o, b_1, and T_d; the solution has an analytical form expressed in terms of the parameters of the DDE in (3.107). One can explicitly determine how the time delay is involved in the solution and, furthermore, how each parameter affects each characteristic root. That enables one to formulate an estimation of time delays in an analytic way. Also, each eigenvalue can be distinguished [58]. The Lambert W function is embedded in various software packages, such as MATLAB.

For first-order scalar DDEs, it has been proven that the rightmost characteristic roots are always obtained by using the principal branch, $k = 0$, and/or $k = -1$ (see Figure 2.16) [75]. For the DDE in (3.107), one has to consider two possible cases for rightmost characteristic roots: characteristic equations of DDEs as in (3.108) can have one real dominant root or two complex

conjugate dominant roots (e.g., see Figure 3.24). Thus, when estimating time delays using characteristic roots, it is required to decide whether it is the former or the latter [58].

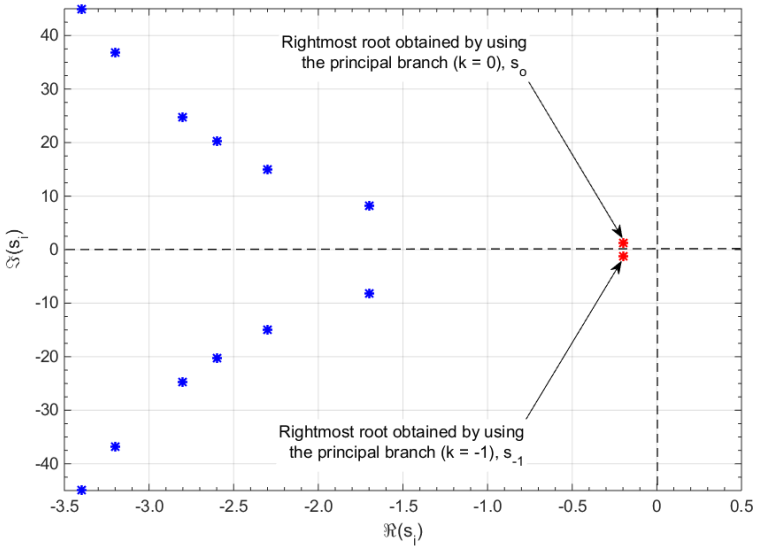


Figure 3.24. Spectrum: the rightmost root is obtained by using the principal branches.

For ODEs, an estimation technique using the logarithmic decrement provides an effective way to estimate ξ [54]. The technique makes use of the form

$$s = -\xi\omega_n \pm j\omega_n\sqrt{1-\xi^2} \tag{3.110}$$

for the determination of s of second-order ODEs. The variables, ξ and ω_n, are obtained from the response of the system, and different approaches can be applied depending on the nature of the response, oscillatory and non-oscillatory [58]. Here, the transient properties, M_o and t_p, for oscillatory response is used to determine ξ and ω_n, see equation (2.15). Then, the drone control system with the unknown T_d, is estimated by the following steps:

Step 1: Calculate ξ and ω_n based on the system altitude response

Step 2: Calculate the 'dominant' roots using $s = -\xi\omega_n \pm j\omega_n\sqrt{(1-\xi^2)}$

Step 3: Equate s to $\frac{1}{T_d}W(T_d a_1 e^{-a_o T_d}) + a_o$, and solve the nonlinear equation for the unknown T_d

Comparing the characteristic equation of the closed-loop system in (3.106) to the first-order system in (3.108), we obtain $b_o = 0$ and $b_1 = -K_P$. The equation in *Step 3* was solved using MATLAB nonlinear solver, *fsolve*.

3.2.4.2 Experimental method. In this approach the transient properties, M_o and t_p, of experiment responses are compared to those of simulation responses for the estimation of T_d.

3.2.5 Design of controllers: first-order model. The effect of T_d on the drone's altitude response was studied and taken into account using analytical, simulation, and experimental approaches in designing the PV and PV-MRAC controllers. The transfer function of the time-delay closed-loop system for the PV controller is given as (see Figure 5.20)

$$\frac{Z(s)}{Z_{des}(s)} = \frac{K_P e^{-sT_d}}{s + (K_P + K_v s)e^{-sT_d}} \tag{3.111}$$

The time-delay system in (3.111) is a neutral type. Comparing the characteristic equation of the closed-loop system in (3.111) to the first-order system in (3.108), we obtain $b_o = 0$ and $b_1 = -(K_P + K_v s)$.

3.2.6 Quadrotor altitude model: black-box approach. MATLAB system identification toolbox App (see Figure 2.11) was used to build models for the drone's altitude motion. Measured data recorded from the drone, using a developed MATLAB/Simulink program (see Figure 5.17), was imported into the App for the modeling process. The input data was the reference vertical speed, $\dot{z}_{ref}(t)$, constrained to $[-1\ \ 1]ms^{-1}$, and the output data was the vertical height, $z(t)$. A P-feedback controller was used to control the drone's altitude motion, with an appropriate K_p value of 1.31.

3.2.6.1 Second-order model. A second-order LTI SISO ARX transfer function model was obtained, which has two poles and one zero, see (3.112). The corresponding state-space model

matrices are: $\mathbf{A} = \begin{bmatrix} -5.8092 & -0.0135 \\ 1.0000 & 0.0000 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{C} = [1.0913 \quad 4.5356]$, and $\mathbf{D} = [0]$. The model has a fit to estimation (best fit) of 91.04%, final prediction error (FPE) of 0.000200948, and mean squared error (MSE) of 0.0001864. Figure 3.25 shows information about the model identification process.

$$\frac{Z(s)}{\dot{Z}_{ref}(s)} = \frac{1.091s + 4.536}{s^2 + 5.809s + 0.01345} \tag{3.112}$$

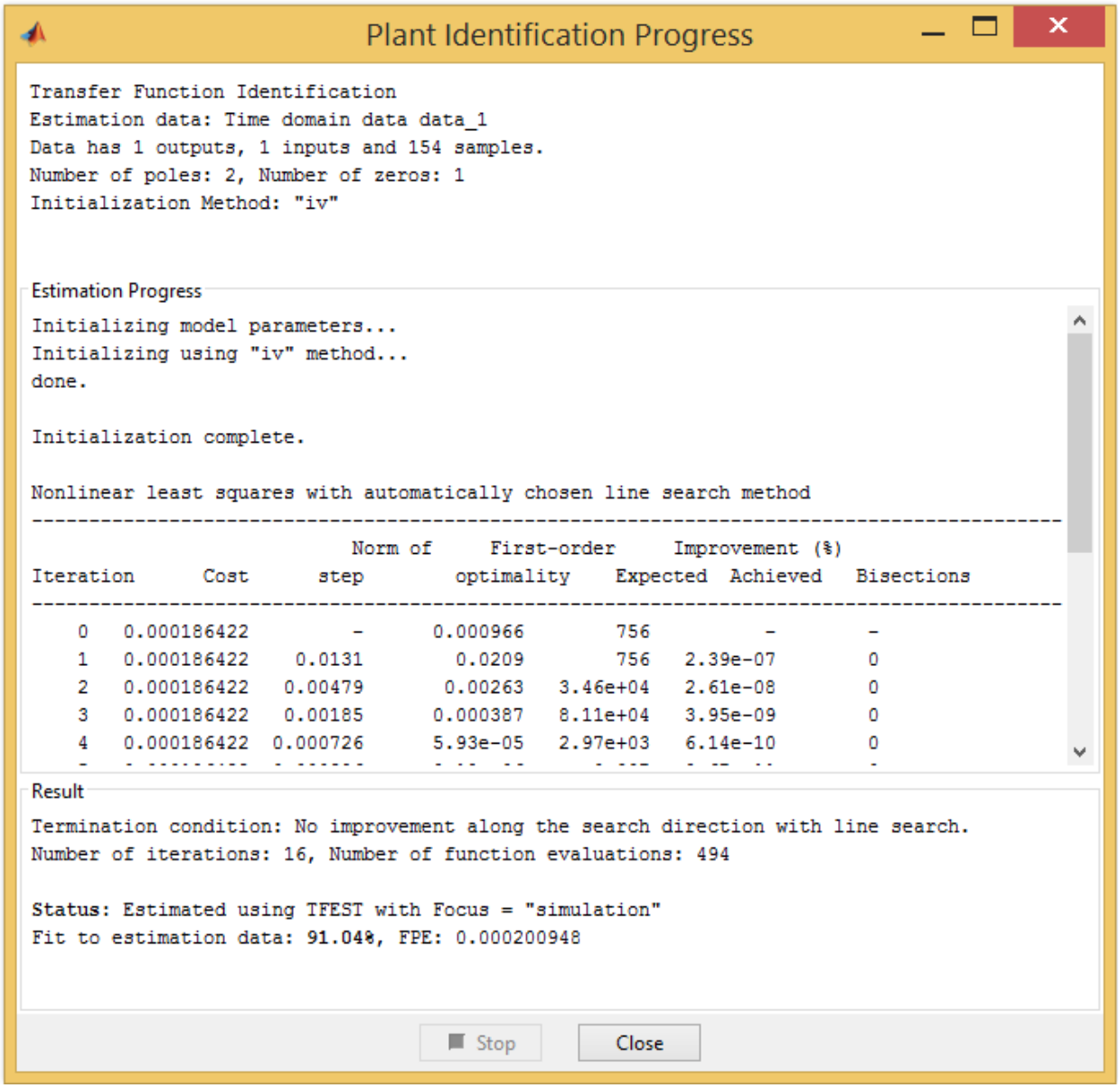


Figure 3.25. MATLAB model identification process information.

The poles of the system in (3.112) are $s = -5.8069$ and -0.0023, which indicates that the system is stable, however, a unity step response of the plant shown in Figure 3.26, shows that the system did not stabilize to the desired value. The controllability matrix, $CONT$, of the system was determined as

$$CONT = \begin{bmatrix} 1.0000 & -5.8092 \\ 0 & 1.0000 \end{bmatrix} \tag{3.113}$$

and observability matrix, $OBSER$, of the system was determined as

$$OBSER = \begin{bmatrix} 1.0913 & 4.5356 \\ -1.8042 & -0.0147 \end{bmatrix} \tag{3.114}$$

The ranks of both $CONT$ and $OBSER$ is 2, which is the dimension of the state matrix, **A**, therefore the system is completely state controllable and completely observable, and thus the system is stabilizable and detectable.

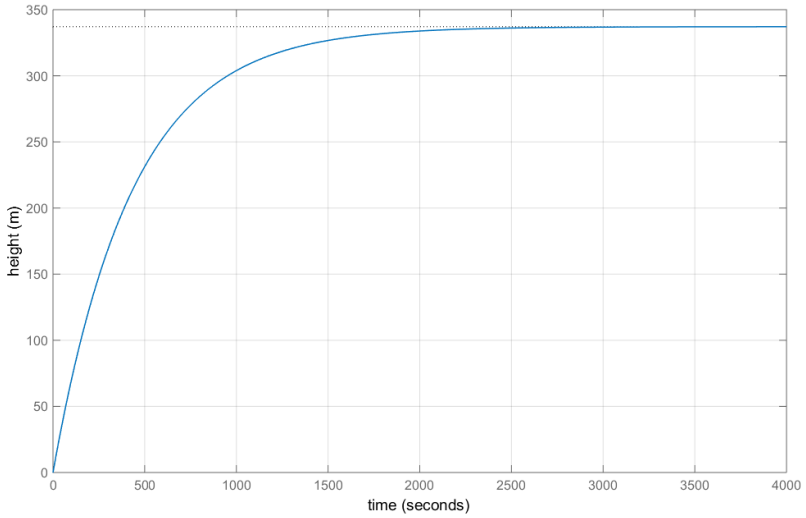


Figure 3.26. Open-loop plant step response: black-box approach (LTI 2nd order model).

3.2.6.2 Second-order model: reduced-form. A first-order transfer function is sufficient to reproduce the dynamics of a DC motor [76]. If the drone's DC motors dynamics are not assumed to be very fast, see Section 3.2.4, then its transfer function can be written as [76]

$$G_{mt}(s) = \frac{\Omega_a(s)}{V_{mt}(s)} = \frac{K_g}{\tau s + 1} \tag{3.115}$$

where τ is the time constant, K_g is the sum of the steady-state gains of the drone's DC motors, $\Omega_a(\mathrm{s})$ is the Laplace transform of the average angular speed of the load shafts, $\omega_a(t)$, and $\mathrm{V}_{mt}(s)$ is the Laplace transform of the average input voltages, $\mathrm{v}_{mt}(t)$ of the motor. Thus, a second-order transfer function for the plant model, see *Section 3.2.4*, can be written as

$$\frac{Z(s)}{\dot{Z}_{ref}(s)} = \left(\frac{K_g}{\tau s + 1}\right)\left(\frac{1}{s}\right) = \frac{K_g}{\tau s^2 + s} \tag{3.116}$$

Currently, the AR.Drone 2.0 does not have sensors attached to measure the angular speeds of the rotors and the input voltages. The research work could have consider the option of attaching sensors to measure the necessary data in the estimation of K_g and τ. For example, a reflective encoder sensors could be placed under the propeller gears to measure the angular position of the load gears. If these sensors data are available, then experimental methods such as the frequency response and bump test could have been used to determine K_g and τ.

In the estimation of K_g and τ, the approach used in *Section 3.2.6.1* was applied, but here 10 samples of recorded data, $(\dot{z}_{ref}(t), z(t))$, were used, and transfer function models, of the form in (3.116), obtained for each data sample. The mean (nominal) values were calculated as $K_{g_n} = 0.9451\ \mathrm{rads}^{-1}\mathrm{V}^{-1}$ and $\tau_n = 0.1304\ s$, and the sample standard deviation values as $s_{K_g} = 0.6895\ \mathrm{rads}^{-1}\mathrm{V}^{-1}$ and $s_\tau = 0.0955\ s$, respectively. Using the *Student's t distribution* and considering only random errors in the measurements, $v = 9, P = 95\%$, and $t_{9,95} = 2.262$, the estimate of the true mean values, based on equation (2.23), for K_g and τ, are given respectively as

$$K_g' = 0.9451 \pm 0.4932 \quad \mathrm{rads}^{-1}\mathrm{V}^{-1} \quad (95\%) \tag{3.117}$$

$$\tau' = 0.1304 \pm 0.0683 \quad \mathrm{s} \quad (95\%) \tag{3.118}$$

Thus, the random uncertainties in the mean values of K_g and τ at 95% confidence due to variation in the measured data set are $\Delta_{K_g} = \pm 0.4932$ and $\Delta_\tau = \pm 0.0683$, respectively. Therefore,

the estimated second order LTI SISO ARX nominal transfer function model for the plant, subject to the random uncertainties in (3.117) and (3.118), is given in equation (3.119). The corresponding state-space model matrices are: $\mathbf{A} = \begin{bmatrix} -7.6687 & 0.0000 \\ 1.0000 & 0.0000 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{C} = [0.0000 \quad 7.2477]$, and $\mathbf{D} = [0]$. The model has an average fit to estimation (best fit) of 86.12%, FPE of 0.002867, and MSE of 0.008252.

$$\frac{Z(s)}{\dot{Z}_{ref}(s)} = \frac{7.2477}{s^2 + 7.6687s} = \frac{0.9451}{0.1304s^2 + s} \tag{3.119}$$

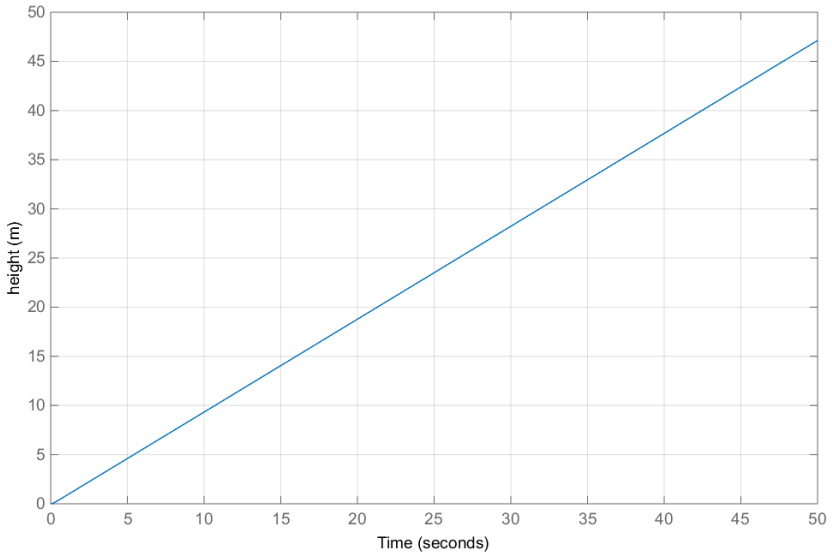


Figure 3.27. Open-loop plant step response: black-box approach (2nd order reduced-model).

The poles of the system in (3.119) are $s = 0.000$ and -7.6687, which indicates that the system is unstable, which is also confirm by the plant's step response shown in Figure 3.27. The controllability matrix, $CONT$, of the system was determined as

$$CONT = \begin{bmatrix} 1.0000 & -7.6687 \\ 0 & 1.0000 \end{bmatrix} \tag{3.120}$$

and observability matrix, $OBSER$, of the system was determined as

$$OBSER = \begin{bmatrix} 0.0000 & 7.2477 \\ 7.2477 & 0.0000 \end{bmatrix} \tag{3.121}$$

The ranks of both $CONT$ and $OBSER$ is 2, which is the dimension of the state matrix, **A**, therefore the system is completely state controllable and completely observable, and thus the system is stabilizable and detectable.

3.2.7 Effects of system uncertainty. As stated in *Section 2.5.3*, a time-delayed P control system, retarded type, is used to investigate the system uncertainty and to measure the stability radii using first-order and second-order plant models developed.

3.2.7.1 First-order model. The transfer function of the time-delayed first-order plant model, see *Sections 3.2.4* and *3.2.6.2*, and equation (3.116), is given as

$$\frac{Z(s)}{\dot{Z}_{ref}(s)} = \frac{K_g e^{-sT_d}}{s} \tag{3.122}$$

where K_g is the uncertain system parameter. Then, the closed-loop transfer function of the P control system is given as

$$\frac{Z(s)}{Z_{des}(s)} = \frac{K_P K_g e^{-sT_d}}{s + K_P K_g e^{-sT_d}} \tag{3.123}$$

Note, in *Section 3.2.4*, in the estimation of T_d, a suitable nominal value of $K_{g_n} = 1.0\ rads^{-1}V^{-1}$ was used, but in this section $K_{g_n} = 0.9451\ rads^{-1}V^{-1}$, obtained in *Section 3.2.6.2*, is used. If K_g is subjected to a certain perturbations, $\pm\Delta_{K_g}$, then (3.123) can be written as

$$\frac{Z(s)}{Z_{des}(s)} = \frac{K_P(K_{g_n} \pm \Delta_{K_g})e^{-sT_d}}{s + K_P(K_{g_n} \pm \Delta_{K_g})e^{-sT_d}} \tag{3.124}$$

From the characteristic equation, we have

$$\dot{z} = -K_P\left(K_{g_n} \pm \Delta_{K_g}\right) z(t - T_d) \tag{3.125}$$

Comparing equations (3.125) and (2.28), we obtain $\mathbf{A} + \mathbf{E}\Delta_{\mathrm{A}}\,\mathbf{F}_{\mathrm{A}} = 0, \mathbf{A} = 0, \mathbf{E}\Delta_{\mathrm{A}}\,\mathbf{F}_{\mathrm{A}} = 0, \mathbf{B} + \mathbf{E}\Delta_{\mathrm{B}}\,\mathbf{F}_{\mathrm{B}} = -K_P\left(K_{g_n} \pm \Delta_{K_g}\right), \mathbf{B} = -K_P K_{g_n}$, and $\mathbf{E}\Delta_{\mathrm{B}}\,\mathbf{F}_{\mathrm{B}} = \pm K_P \Delta_{K_g}$. In the computation of the

real stability radius, $r_{\Re}$, the perturbation scaling parameters, $\mathbf{E}$, $\mathbf{F}_{\mathrm{A}}$, and $\mathbf{F}_{\mathrm{B}}$ can be taken to be the appropriate identity matrix. Thus, for an unstructured case where $m = p = n = 1$, $\mathbf{E} = \mathbf{F}_{\mathrm{A}} = \mathbf{F}_{\mathrm{B}} = 1$ is used, which gives $\Delta_{\mathrm{B}} = \pm K_P \Delta_{K_g}$. From the statistical-experimental results in *Section 3.2.6.2*, $\Delta_{\mathrm{B}} = \pm(1.31)(0.4932) = \pm 0.6461$.

3.2.7.2 Second-order model: reduced-form. Using the transfer function of the plant in equation (3.116), the time-delayed plant model transfer function can be written as

$$\frac{Z(s)}{\dot{Z}_{ref}(s)} = \frac{K_g e^{-sT_d}}{\tau s^2 + s} \tag{3.126}$$

and the corresponding closed-loop transfer function of the P control system as

$$\frac{Z(s)}{Z_{des}(s)} = \frac{K_P K_g e^{-sT_d}}{\tau s^2 + s + K_P K_g e^{-sT_d}} \tag{3.127}$$

where K_g and τ have nominal values of $K_{g_n} = 0.9451\, rads^{-1}V^{-1}$ and $\tau_n = 0.1304\, s$, respectively, obtained in *Section 3.2.6.2*. If K_g and τ are subjected to a certain perturbations, $\pm\Delta_{K_g}$ and $\pm\Delta\tau$, respectively, then (3.127) can be written as

$$\frac{Z(s)}{Z_{des}(s)} = \frac{K_P(K_{g_n} \pm \Delta_{K_g}) e^{-sT_d}}{(\tau_n \pm \Delta_\tau)s^2 + s + K_P(K_{g_n} \pm \Delta_{K_g}) e^{-sT_d}} \tag{3.128}$$

From the characteristic equation, we have

$$(\tau_n \pm \Delta_\tau)\ddot{z} + \dot{z} + K_P\left(K_{g_n} \pm \Delta_{K_g}\right) z(t - T_d) = 0 \tag{3.129}$$

Writing (3.129) in state-space representation, we have

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & \{-1/(\tau_n \pm \Delta_\tau)\} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 & 0 \\ \left\{-K_P\left(K_{g_n} \pm \Delta_{K_g}\right)/(\tau_n \pm \Delta_\tau)\right\} & 0 \end{bmatrix} \mathbf{x}(t - T_d) \tag{3.130}$$

where $\mathbf{x}(t) = [\, z \;\; \dot{z}]^T$. Comparing equations (3.130) and (2.28), we obtain $\mathbf{A} + \mathbf{E}\Delta_{\mathrm{A}}\, \mathbf{F}_{\mathrm{A}} = \begin{bmatrix} 0 & 1 \\ 0 & \{-1/(\tau_n \pm \Delta_\tau)\} \end{bmatrix}$, $\mathbf{B} + \mathbf{E}\Delta_{\mathrm{B}}\, \mathbf{F}_{\mathrm{B}} = \begin{bmatrix} 0 & 0 \\ \left\{-K_P\left(K_{g_n} \pm \Delta_{K_g}\right)/(\tau_n \pm \Delta_\tau)\right\} & 0 \end{bmatrix}$, $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -1/\tau_n \end{bmatrix}$,

and $\mathbf{B} = \begin{bmatrix} 0 & 0 \\ -K_P K_{g_n}/\tau_n & 0 \end{bmatrix}$. In the computation of $r_{\Re}$, the perturbation scaling parameters, $\mathbf{E} = \mathbf{F}_\mathrm{A} = \mathbf{F}_\mathrm{B} = \mathrm{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ is used, and therefore, the values of Δ_A and Δ_B are computed as

$$\Delta_\mathrm{A} = \mathbf{E}\Delta_\mathrm{A}\,\mathbf{F}_\mathrm{A} = \begin{bmatrix} 0 & 0 \\ 0 & \{1/\tau_n - 1/(\tau_n \pm \Delta_\tau)\} \end{bmatrix} \tag{3.131}$$

$$\Delta_\mathrm{B} = \mathbf{E}\Delta_\mathrm{B}\,\mathbf{F}_\mathrm{B} = \begin{bmatrix} 0 & 0 \\ \left\{K_P K_{g_n}/\tau_n - K_P\left(K_{g_n} \pm \Delta_{K_g}\right)/(\tau_n \pm \Delta_\tau)\right\} & 0 \end{bmatrix} \tag{3.132}$$

Thus, from the statistical-experimental results in *Section 3.2.6.2*, see equations (3.117) and (3.118), and using equations (3.131) and (3.132), sample of possible values of Δ_A and Δ_B can be computed as shown in Table 3.1.

Table 3.1

Possible Values of Δ *from Experiments: Second-order Reduced-model*

	Case 1	Case 2	Case 3
	$\Delta_{K_g} = 0.0100$ & $\Delta_\tau = 0.0000$	$\Delta_{K_g} = 0.0100$ & $\Delta_\tau = -0.0010$	$\Delta_{K_g} = -0.2000$ & $\Delta_\tau = -0.0100$
$\Delta_\mathrm{A} = \mathbf{E}\Delta_\mathrm{A}\mathbf{F}_\mathrm{A}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & -0.059 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & -0.637 \end{bmatrix}$
$\Delta_\mathrm{B} = \mathbf{E}\Delta_\mathrm{B}\mathbf{F}_\mathrm{B}$	$\begin{bmatrix} 0 & 0 \\ -0.101 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ -0.175 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 1.388 & 0 \end{bmatrix}$
$\Delta = [\Delta_\mathrm{A}, \Delta_\mathrm{B}]$	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -0.101 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -0.059 & -0.175 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -0.637 & 1.388 & 0 \end{bmatrix}$

3.2.7.3 Effects of disturbance rejection: quadrotor payload. The effects of the dynamic load disturbances introduced by increasing the payload on the drone's altitude response were experimented using the designed PV and PV-MRAC flight control systems. Also, the stability bounds within which the changing mass-inertia parameters of the system due to acquired object

not destabilize the drone were determined. Additionally, it was demonstrated experimentally the stability behavior of the drone undergoing instantaneous step payload changes.

CHAPTER 4

Simulation and Experimental Setups: Ground Robot

This chapter presents and discusses all the simulation and experimental setups used for the research work on the ground robot. The developed Simulink model shown in Figure 4.1 was used to implement the kinematic model of the unicycle in equation (3.2).

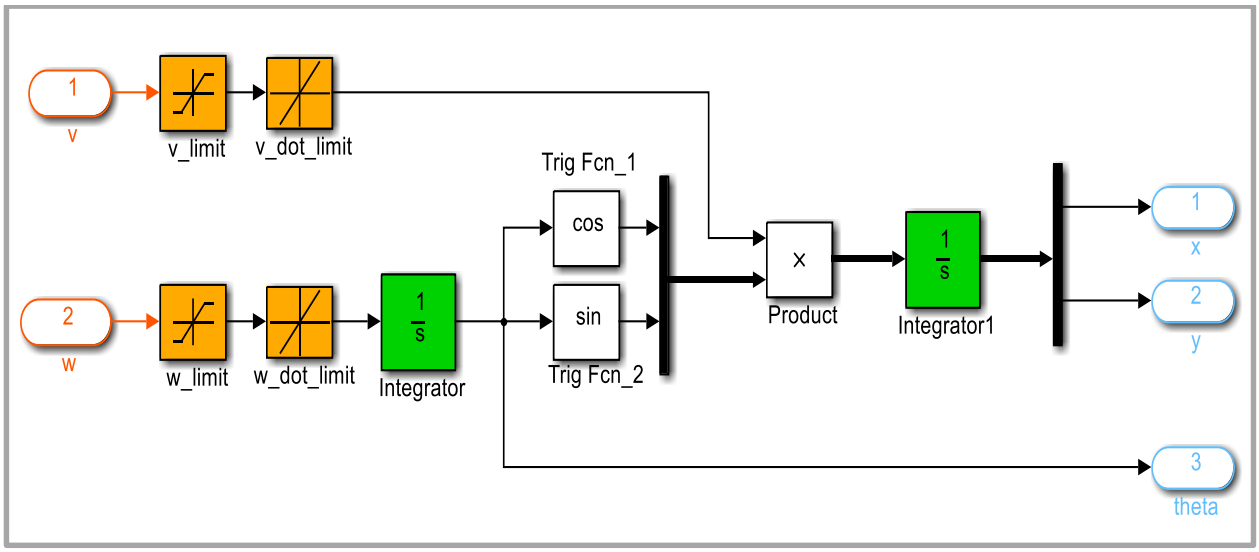


Figure 4.1. Simulink model for the unicycle.

4.1 Control Algorithms for Individual Behaviors

Figures 4.2 to 4.5 below shows the Simulink models used to simulate the individual DDWMR behaviors control algorithms obtained in *Section 3.1.2.1*.

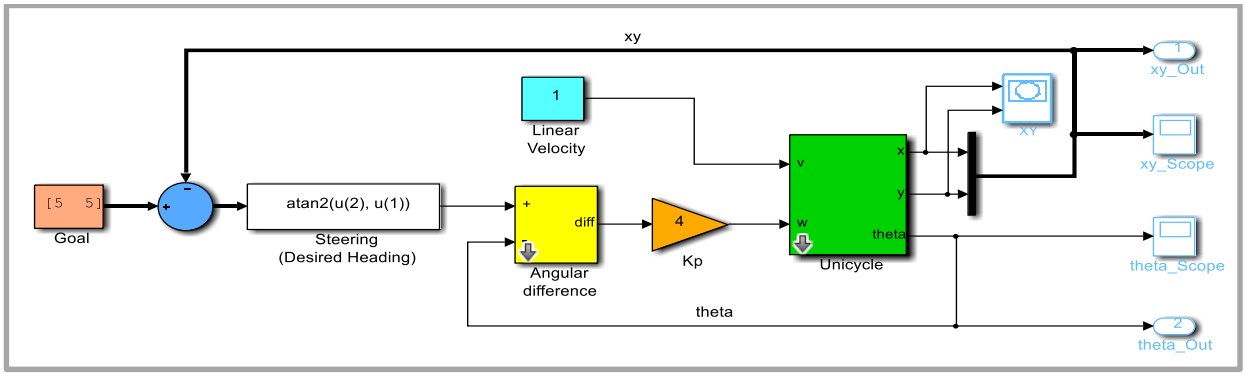


Figure 4.2. Simulink model that drives the robot to a point.

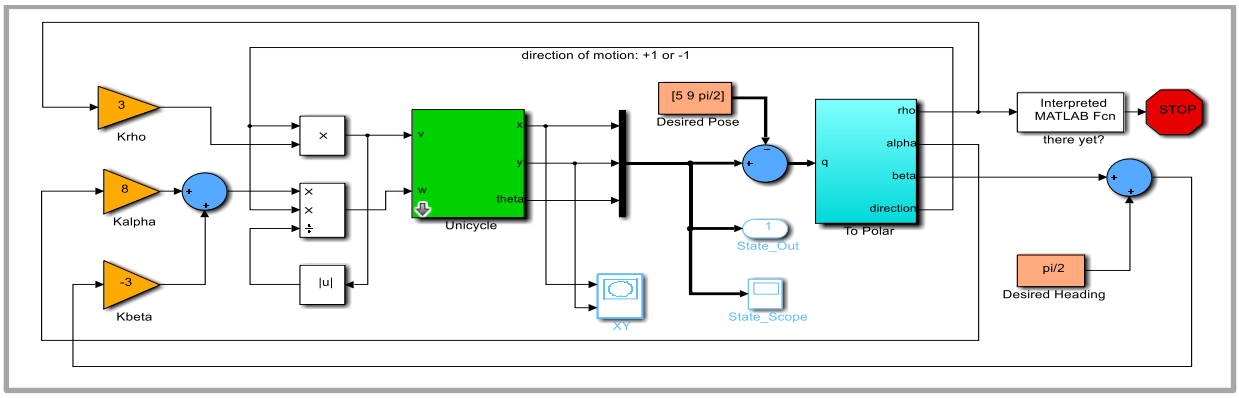


Figure 4.3. Simulink model that drives the robot to a pose.

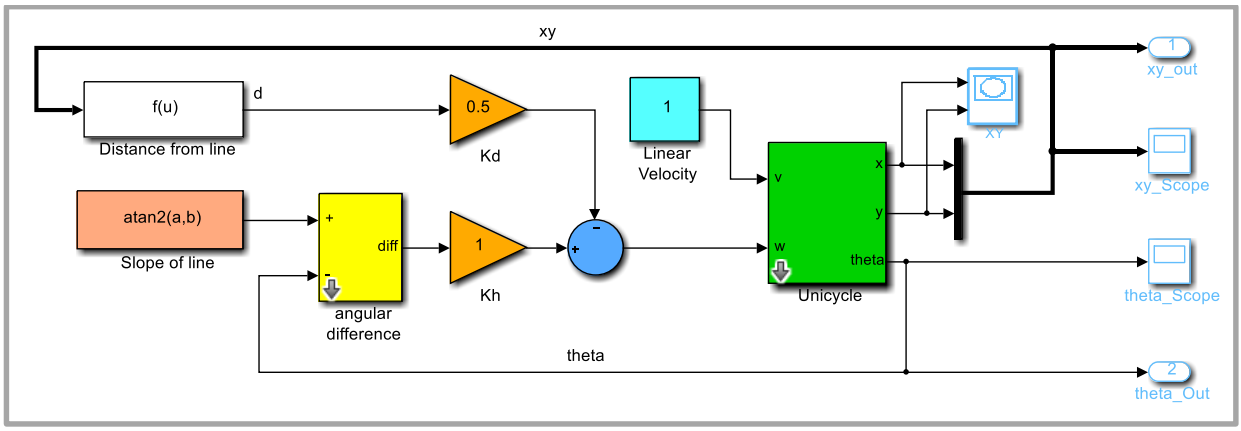


Figure 4.4. Simulink model that drives the robot to follow a line.

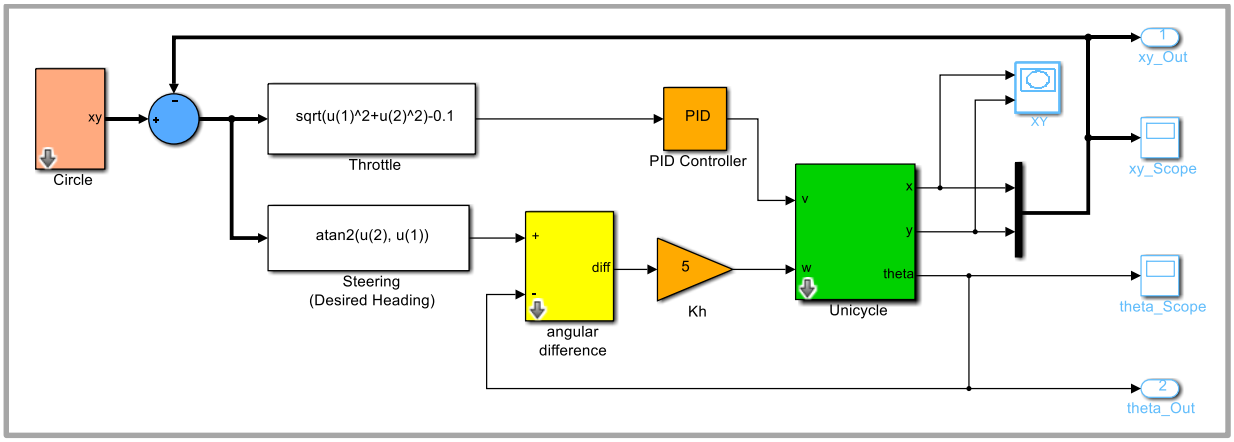


Figure 4.5. Simulink model that drives the robot to follow a circle.

4.2 Control Algorithm for Navigation System

4.2.1 Simulations. A developed MATLAB robot simulator was used to simulate the navigation control algorithm presented in *Section 3.1.2.2.5*. The control algorithm combines the GTG, AO, and FW controllers into a full navigation system for the robot. The MATLAB algorithm that controls the robot simulator implements *finite state machine (FSM)* to solve the full navigation problem. The FSM uses a set of *if/elseif/else* statements that first check which state (or behavior) the robot is in, and then based on whether an event (condition) is satisfied, the FSM switches to another state, or stays in the same state, until the robot reaches its goal. Figure 4.6 below shows a sequence of movement of the MATLAB robot simulator implementing the navigation system. The robot navigates around a cluttered, complex environment without colliding with any obstacles and reaching its goal location successfully.

4.2.2 Experiments. The control algorithm built for the navigation architecture presented in *Section 3.1.2.2.5* has been experimented on Dr Robot X80SV, programmed using MATLAB GUI, in an office environment. The X80SV can be made to move to a goal while avoiding obstacles along the way. The X80SV is a fully wireless networked that uses two quadrature encoders on each wheel for measuring its position, and seven IR and three ultrasonic range sensors for collision detection.

It has 2.6x high resolution Pan-Tilt-Zoom CCD camera with two-way audio capability, two 12V motors with over 22kg.cm torque each, and two pyroelectric human motion sensors. It has a dimension of 38cm (length) x 35cm (width) x 28cm (height), maximum payload of 10kg (optional 40kg) with robot weight of 3kg. Its 12V 3700mAh battery pack has three hours nominal operation time for each recharging, and can drive up to a maximum speed of 1.0ms^{-1}. The distance between the wheels is 26cm and the radius of the wheels is 8.5cm.

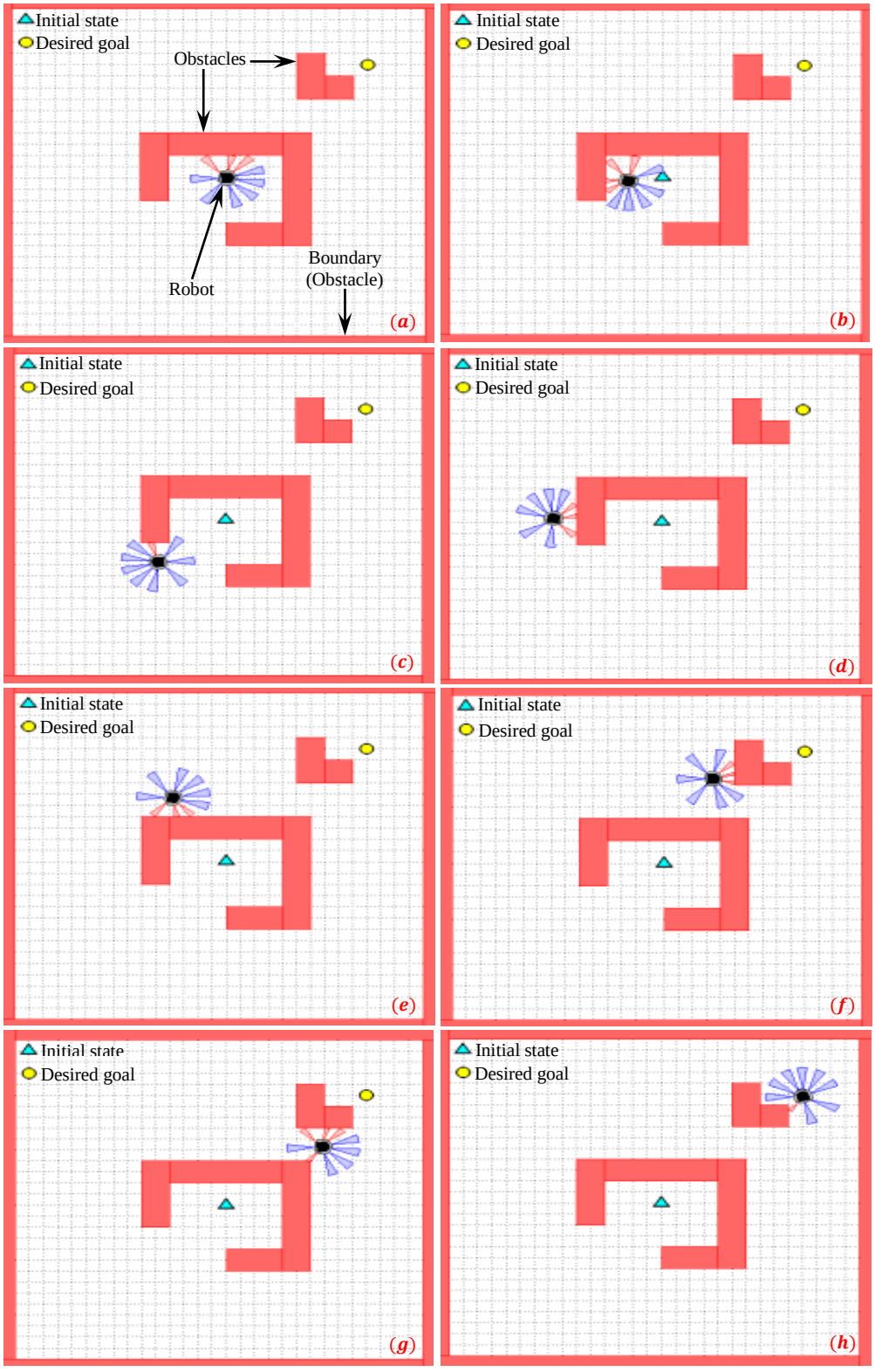


Figure 4.6. Sequence of the MATLAB robot simulator implementing the navigation system.

The PID-feedback system depicted in Figure 4.7 shows how the DC motor system of the robot is controlled. Figure 4.8 shows the setup used for the experiments. After a connection is established between the host PC and the robot through the wireless router, the X80SV control program receives and sends the motion/sensors signals using *ActiveX control*. Also, the program directly exchange multimedia data with the Pan-Tilt-Zoom camera through an *ActiveX control*.

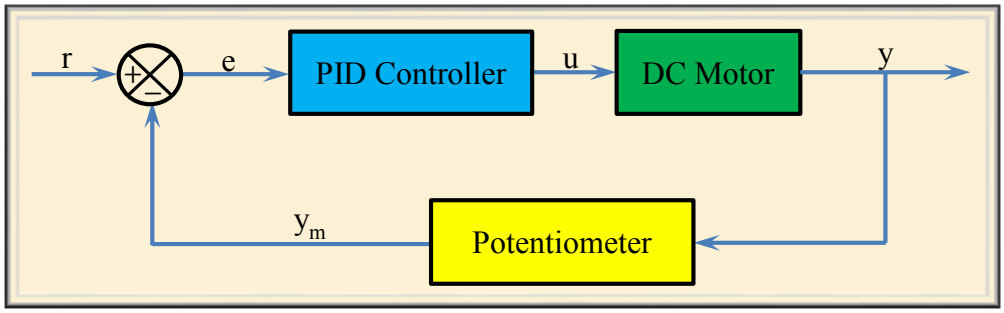


Figure 4.7. Closed-loop feedback system for controlling the DC motors of the X80SV.

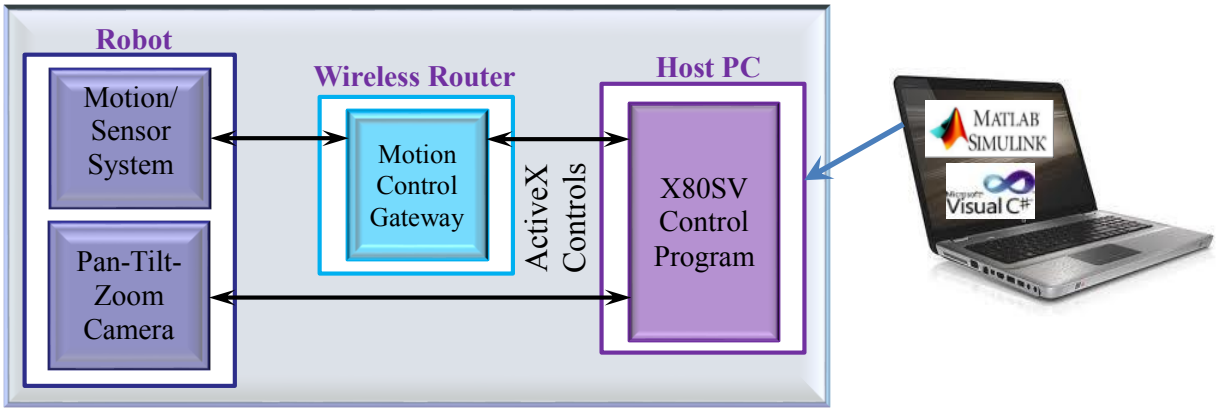


Figure 4.8. Setup used for the X80SV experiments.

A screenshot of the MATLAB interface developed for the X80SV control is shown in Figure 4.9. The interface was developed by mimicking a similar interface developed in C# by the manufacture of the robot, Dr Robot Inc (see Figure 1.6). The motivation for using MATLAB instead of building upon the provided C# interface was to take advantage of the ease of simulation,

quick and ease of developing GUI, and making use of the in-built control strategies libraries in MATLAB for this research and future studies.

The MATLAB interface has three sections: information about the robot settings and sensors, multimedia, and the vision and control. The robot settings information includes the IP addresses of the robot and the camera, the robot wheel radius, distance between the robot wheels, and the encoder count per revolution. The sensors information, updated in real-time, includes the infrared (IR), ultrasonic (US), motor, human, temperature, battery, and the position of the robot.

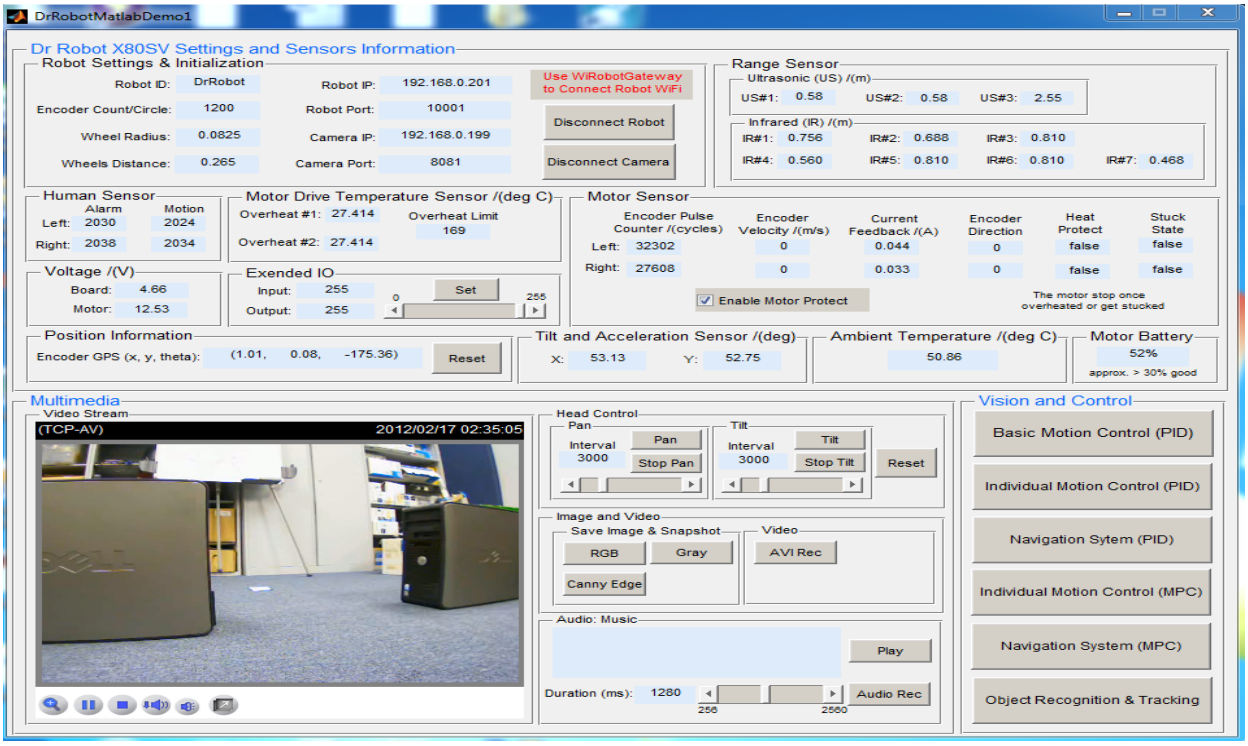


Figure 4.9. MATLAB X80SV control program interface.

The multimedia section include real-time video stream from the robot, which can be controlled using a pan and tilt tools. Also, the section has tools for capturing images and recording the video stream. In addition, the section also has a tool to capture live audio from the robot. The vision and control section has tools for performing ‘Basic Motion Control’, ‘Individual Motion Control (PID and MPC)’, ‘Navigation System (PID and MPC), and ‘Object Recognition and

Tracking'. The MPC (model predictive control) and the 'Object Recognition and Tracking' tools are for future research work.

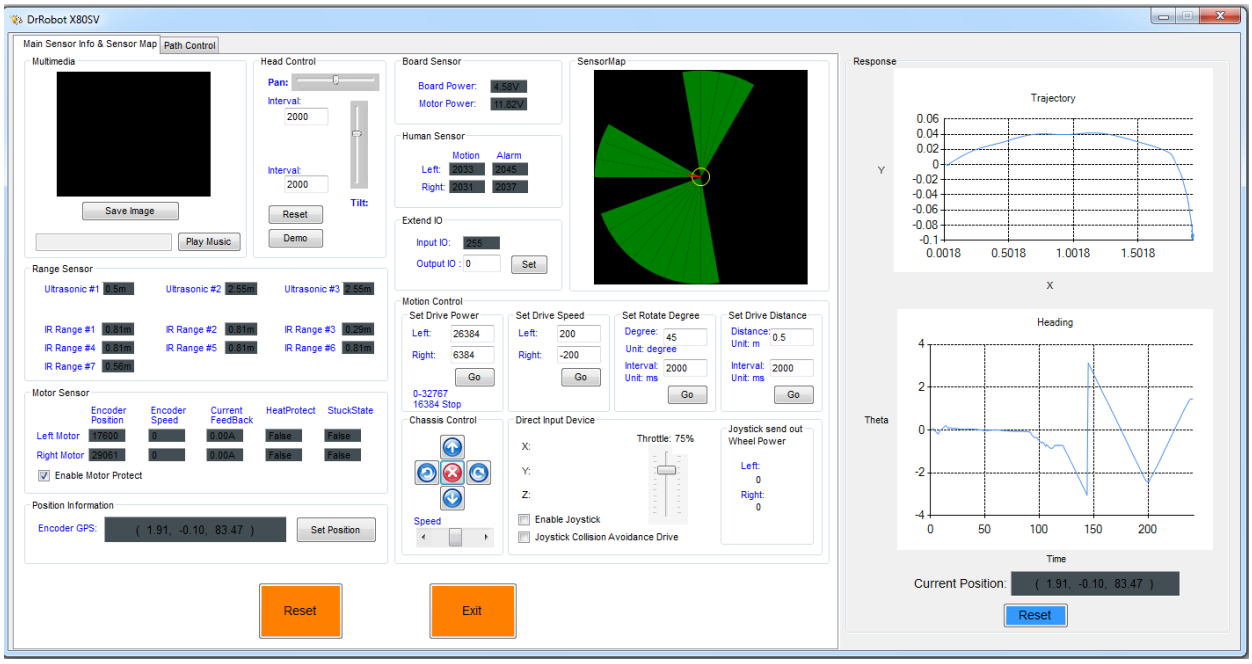


Figure 4.10. Modified C# X80SV program: main sensor information and sensor map.

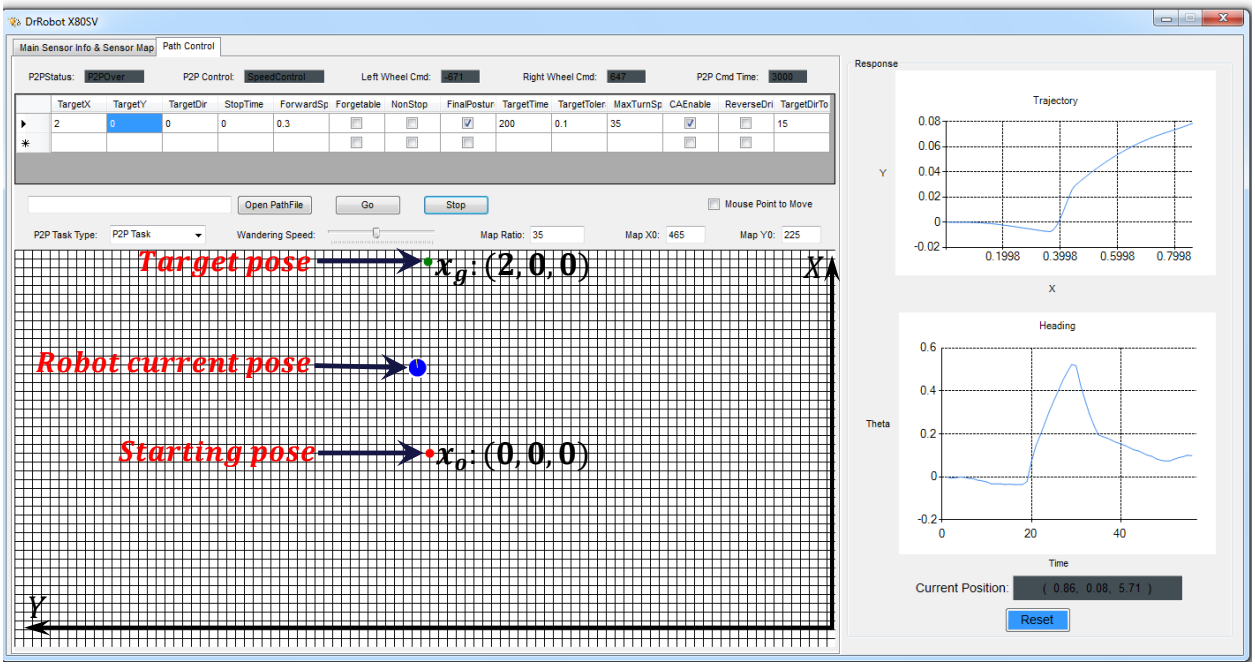


Figure 4.11. Modified C# X80SV program: path control.

The ‘Basic Motion Control’ makes the robot perform operations such as move forward, backward, or rotate, etc. The ‘Individual Motion Control (PID)’ makes the robot ‘move to a point’, ‘move to a pose’, ‘follow a path’, and ‘avoid obstacles. The ‘Navigation System (PID)’ makes the robot to move to a goal in the presence of obstacles.

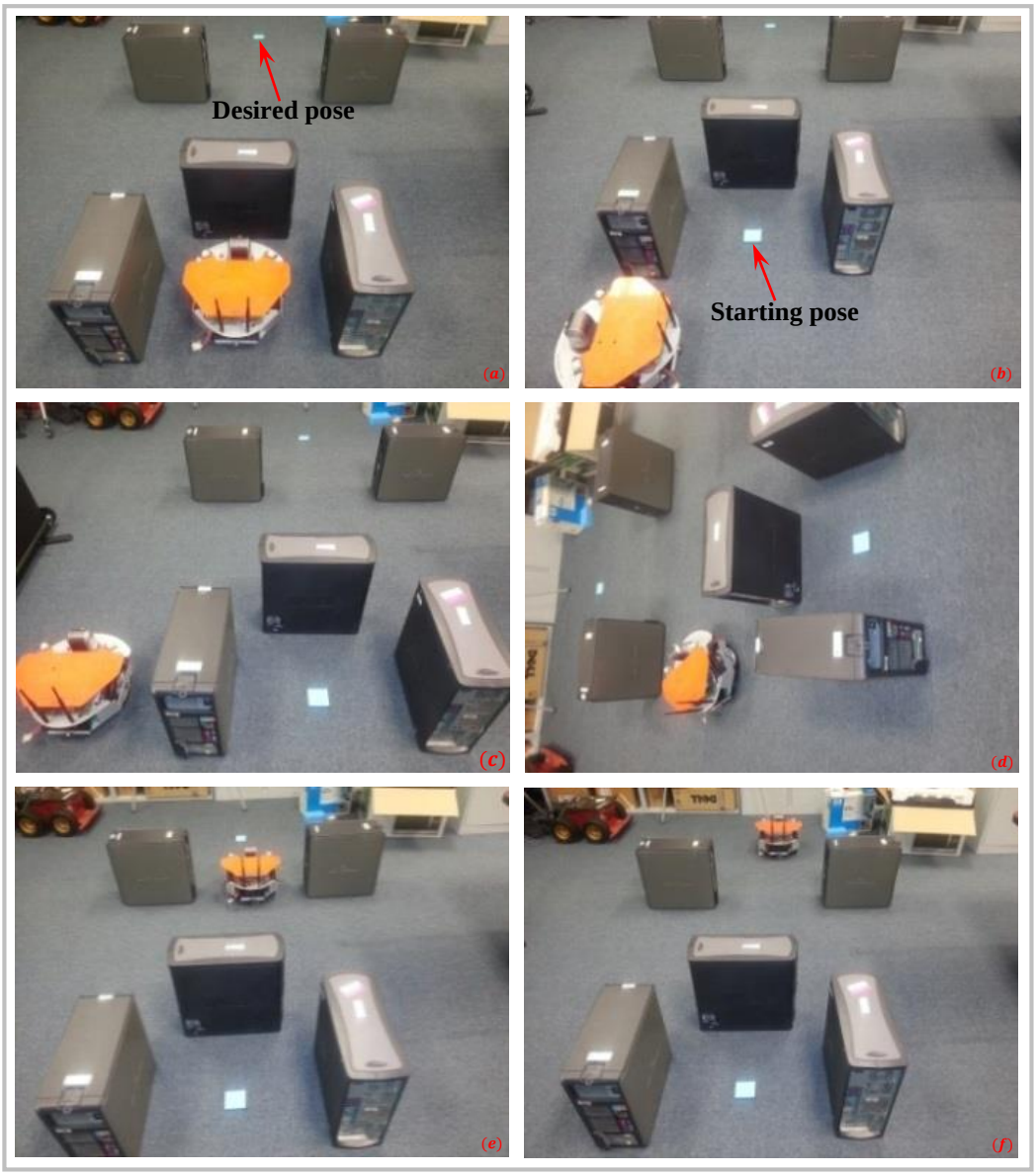


Figure 4.12. Experimental setup showing sequence of the Dr Robot X80SV movement.

Due to the difficulty in sending signals to the DC motors and receiving measured data using the MATLAB GUI program, in real time, the C# program (see Figure 1.6) was modified for the

implementation of the control algorithms. The modified C# GUI program has two interfaces: ‘Main Sensor info and Sensor Map’ and ‘Path Control’. The ‘Main Sensor info and Sensor Map’ interface, see Figure 4.10, contain similar tools just like the MATLAB GUI program.

The ‘Path Control’, see Figure 4.11, is the interface used to implement the control algorithms. It contains the option of choosing to implement the individual behavior or the navigation system control algorithms. Also, it has tools for making settings such as: target pose, stop time, forward speed, stopping tolerances (for x, y, and φ), etc. Additionally, the interface has a 2D grid to help visualize, in real time, the robot’s movement. Moreover, there is a real-time plot of the robot’s trajectory and heading. Figure 4.12 shows an experimental setup showing a sequence of movement of the X80SV implementing the navigation control algorithm.

CHAPTER 5

Simulation and Experimental Setups: Aerial Robot

This chapter presents and discusses all the simulation and experimental setups used for the research work on the aerial robot.

5.1 Control of Quadrotor Motions: White-box Approach (Nonlinear Model)

The simulations were carried out using MATLAB/Simulink models to design and validate the control algorithms discussed in *Section 3.2.1.2*. Simulink block, *quadrotor dynamics*, shown in Figure 5.1, was used to implement the dynamics and kinematics of the quadrotor discussed in *Section 3.2.1.1*. The block employs a MATLAB *S-function* to generate a continuous state output, **x**.

S-functions (system-functions) provide a powerful mechanism for extending the capabilities of the Simulink environment. *S-functions* follow a general form and can accommodate continuous, discrete, and hybrid systems. An algorithm in an *S-function* is implemented by following a set of simple rules, and through an *S-function block* it is added to a Simulink model. The block consists of a set of inputs, a set of states, and a set of outputs, where the outputs are functions of the simulation time, the inputs, and the states.

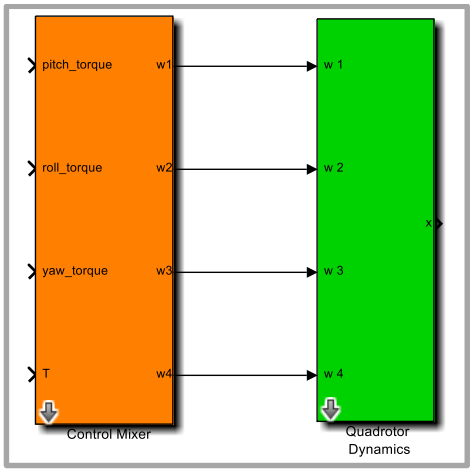


Figure 5.1. Simulink model for the quadrotor dynamics and control mixer blocks.

The *quadrotor dynamic block* is connected to the *control mixer block* (see Figure 5.1), whose inputs are the three torques and the total thrust acting on the airframe. The *control mixer block* is used to compute the four rotor speeds, which serves as inputs to the *quadrotor dynamics block*. These blocks were developed based on the models introduced in [39].

5.1.1 Simulink models for the control algorithms. This section discusses various Simulink models developed for the different control algorithms presented in *Section 3.2.1.2*.

5.1.1.1 Altitude controller. The Simulink model shown in Figure 5.2 implements the altitude control algorithm. For a given desired height, z^*, the altitude control loop, based on (3.63), is used to generate the required thrust, T, while the torque, $\boldsymbol{\Gamma}_B$, to the airframe is set to zero.

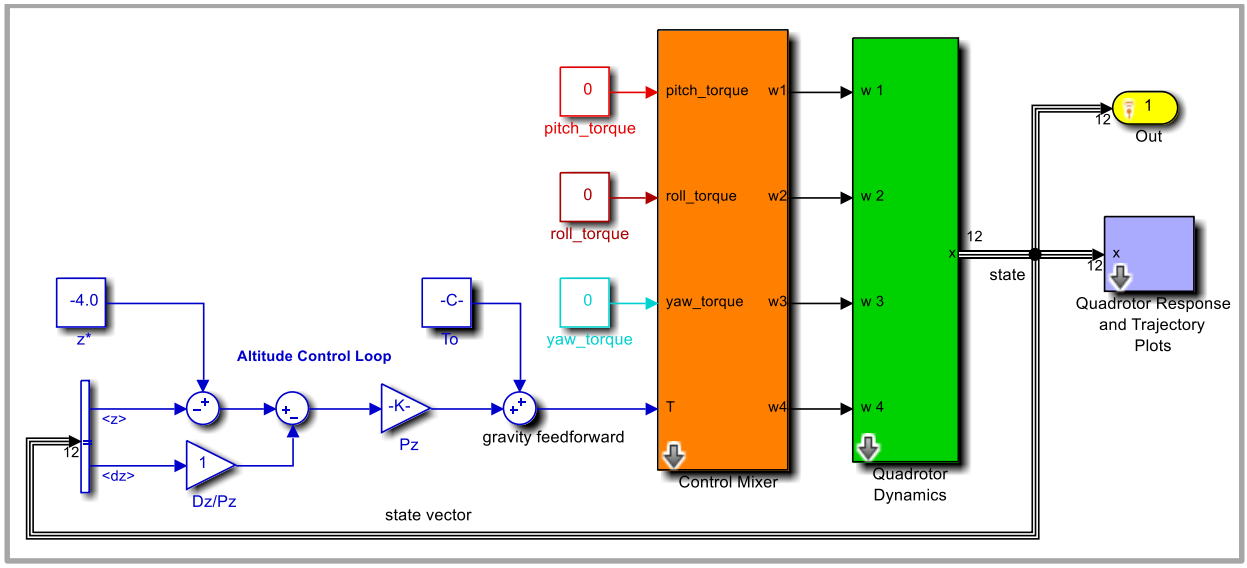


Figure 5.2. Simulink model for altitude control.

5.1.1.2 Yaw controller. The Simulink model shown in Figure 5.3 implements the yaw control algorithm. The vehicle must first move to a desired height before yawing, hence the altitude controller is combined with the yaw controller, with an *if-else control block* to implement the conditional statement. Thus, for a desired yaw, yaw^*, the altitude control loop drives the quadrotor

to a desired height, z^*, and based on (3.62), the yaw control loop is used to generate the required yaw torque, τ_{φ_B}, while keeping the other two torques on the airframe at zero.

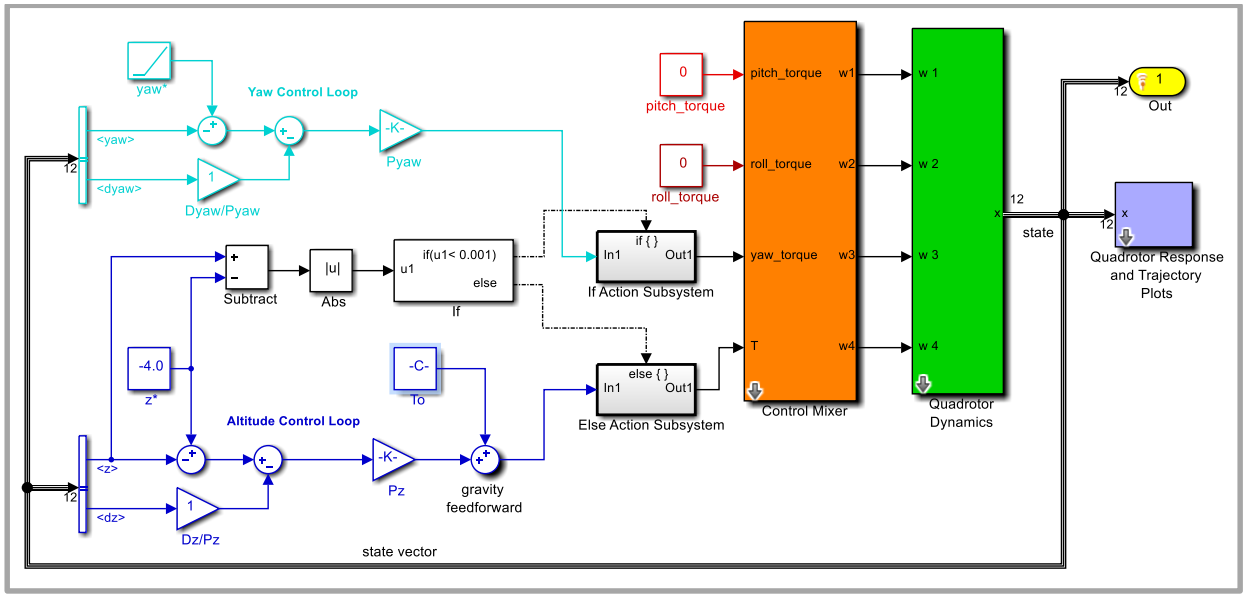


Figure 5.3. Simulink model for yaw control.

5.1.1.3 Pitch controller. The Simulink model shown in Figure 5.4 implements the pitch control algorithm. The vehicle must first move to a desired height before pitching, hence the altitude controller is combined with the pitch controller, with an *if-else control block* to implement the conditional statement. Thus, for a desired pitch, $pitch^*$, the altitude control loop drives the quadrotor to a desired height, z^*, and based on (3.60), the pitch control loop is used to generate the required pitch torque, τ_{θ_B}, while keeping the other two torques on the airframe at zero.

5.1.1.4 Roll controller. The Simulink model shown in Figure 5.5 implements the roll control algorithm. The vehicle must first moved to a desired height before rolling, hence the altitude controller is combined with the roll controller, with an *if-else control block* to implement the conditional statement. Thus, for a desired roll, $roll^*$, the altitude control loop drives the quadrotor to a desired height, z^*, and based on (3.61), the roll control loop is used to generate the required roll torque, $\tau_{\emptyset_B}$, while keeping the other two torques on the airframe at zero.

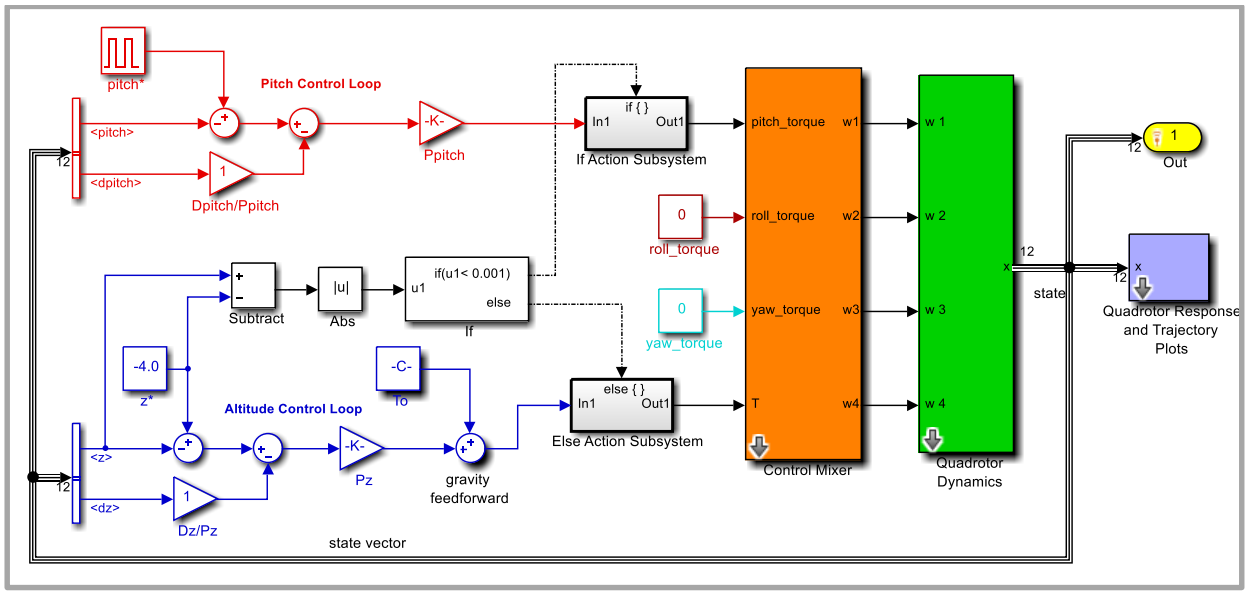


Figure 5.4. Simulink model for pitch control.

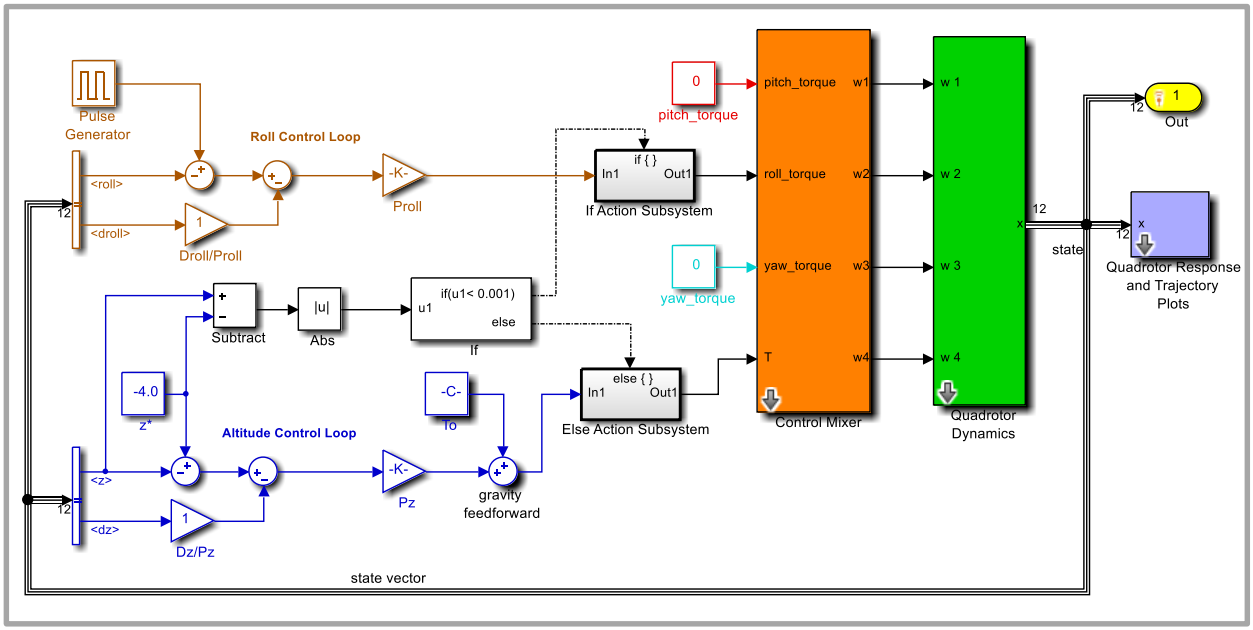


Figure 5.5. Simulink model for roll control.

5.1.1.5 Position controller. The Simulink model shown in Figure 5.6 below implements the position controller. The controller combines the x and y directions control algorithms based on (3.73), with the altitude controller, to drive the quadrotor to the desired position, ($x^*, y^*,$ and z^*). The yaw angle, φ, is also needed, hence the inclusion of the yaw controller.

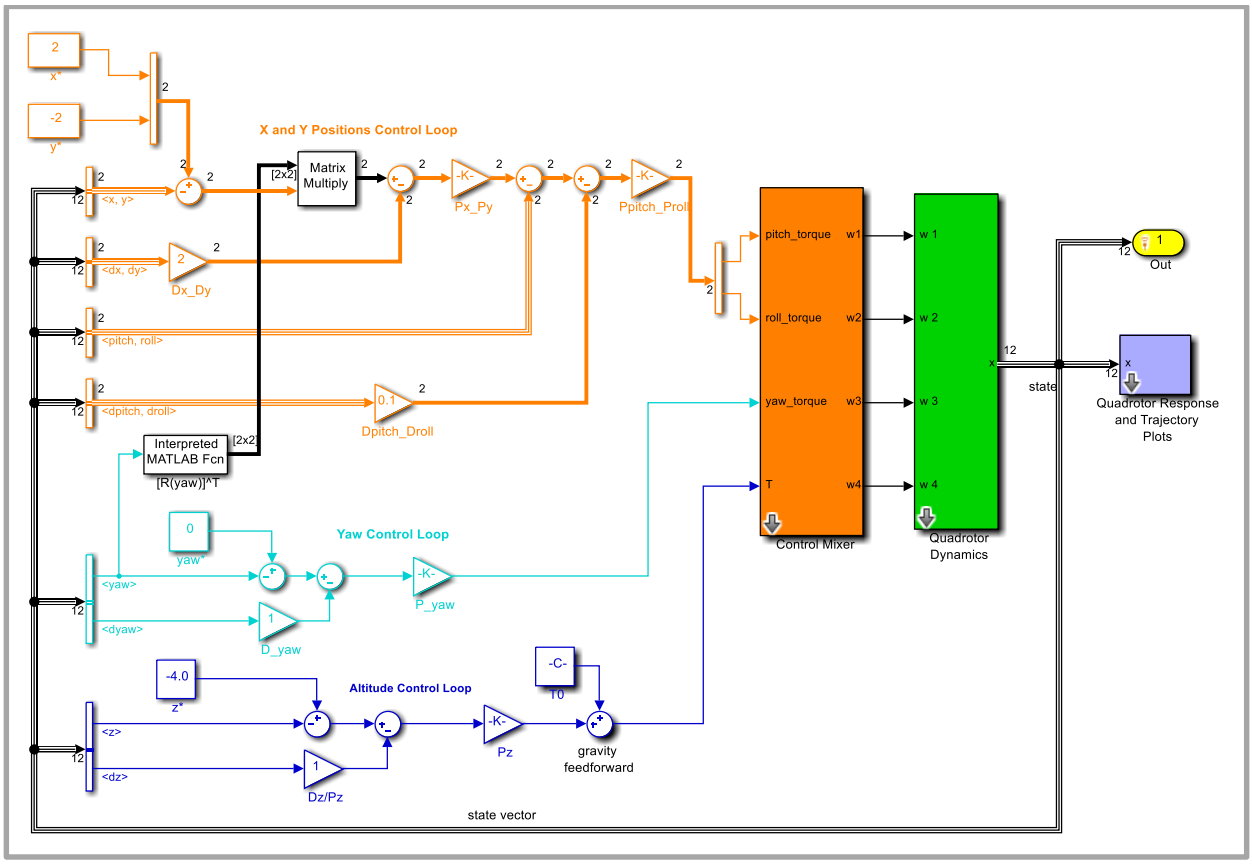


Figure 5.6. Simulink model for position control.

5.1.2 MATLAB GUI for the simulations. A screenshot of a robust and interactive MATLAB GUI interface developed to implement the various control algorithms simulations discussed above is shown in Figure 5.7.

The GUI interface has five sections: initialization and information about the quadrotor settings, selection of controller type and its parameter settings, displaying of the quadrotor current attitude and position values, displaying of some performance index values of a simulation run, and plotting of the quadrotor's response and trajectory. The interface works by first initializing the quadrotor's parameters, and then selecting the type of controller and its parameter settings required for the simulation. The selected controller can then simulated using its parameter settings selected. The quadrotor attitude and position real-time state values and responses can be displayed, and the

3D trajectory of the quadrotor can also be visualized. An animated quadrotor, which was graphically designed, receives data from the simulation, and execute the response in real time. After the simulation some performance index values are displayed.

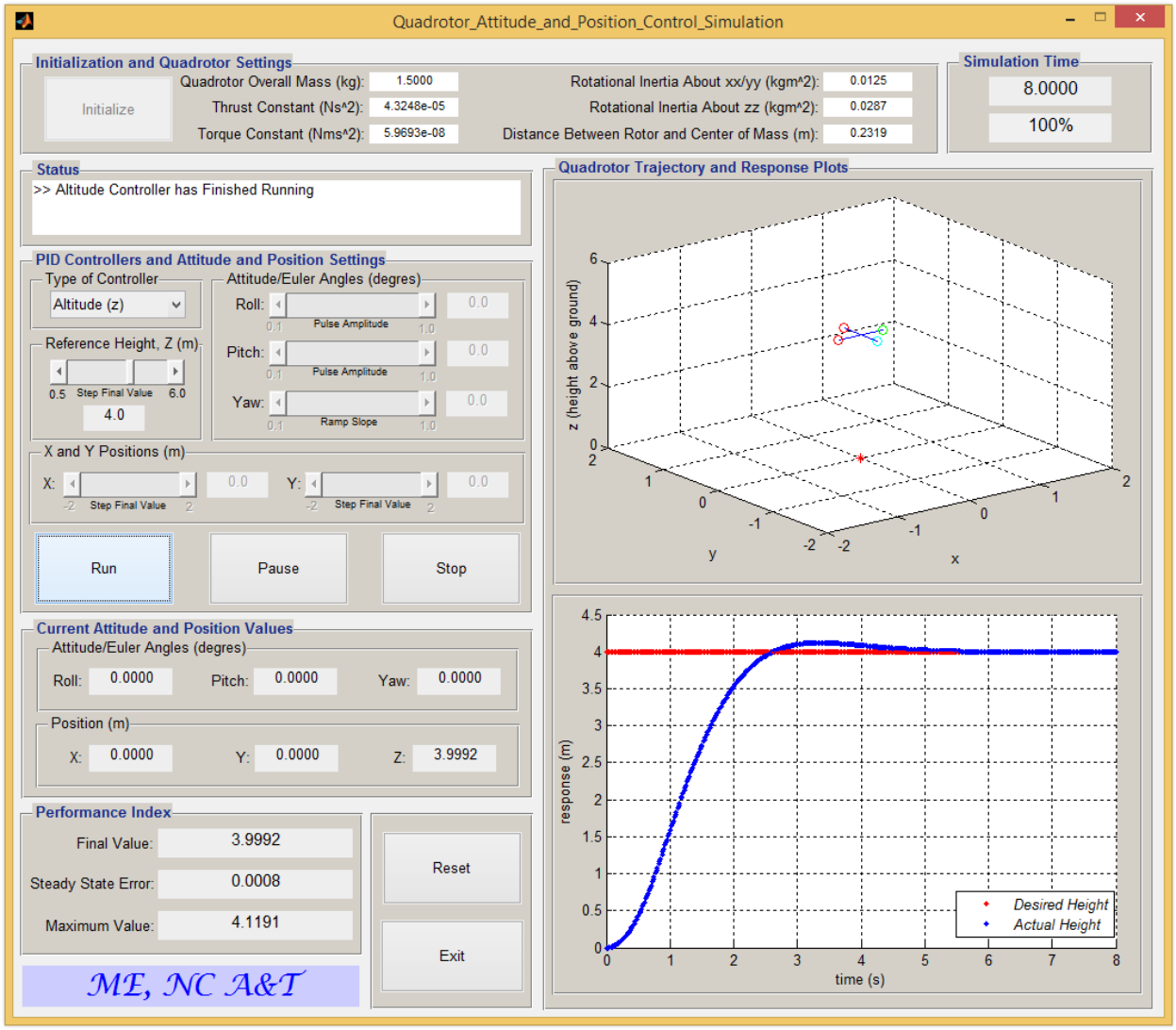


Figure 5.7. MATLAB GUI used to implement the various control algorithm simulations.

The quadrotor real-time response and trajectory plots, shown in the GUI, were achieved by using *S-function block* in the Simulink models. The quadrotor real-time attitude and position state values, displayed in the GUI, were also achieved by using MATLAB *Output block run-time object*

and an *event-listener* mechanism *add_exec_event_listener* command, and then synchronizing the *run-time object* with the Simulink execution. The state output, **x**, is the input to the two blocks.

5.2 Control of Quadrotor Altitude Motion: White-box Approach (Linearized Model)

This section presents the Simulink models and experimental setups for the autonomous altitude control of quadrotor and the drone using the altitude linearized model.

5.2.1 Simulations. The Simulink model in Figure 5.8 was used to model the state-space representation of the plant, and to simulate a unit-step response to confirm the stability of the plant.

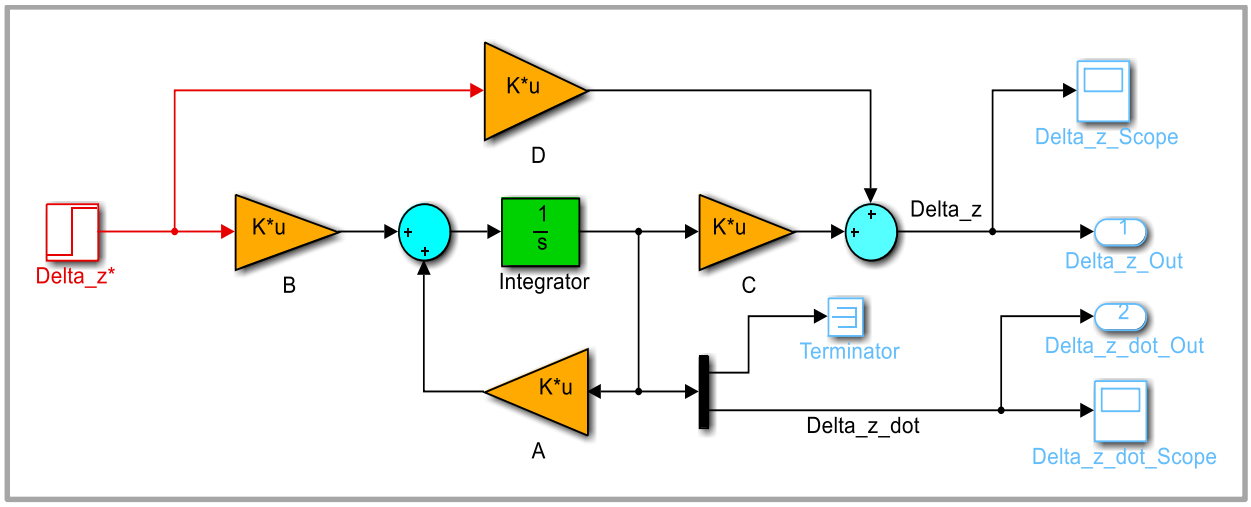


Figure 5.8. Simulink model for open-loop plant.

5.2.1.1 Full-state feedback without nonlinear effects. Figure 5.9 shows the Simulink model used to simulate a unit-step response of the closed-loop system for the full-state feedback control system without the nonlinear effects.

5.2.1.2 Pole placement with saturation effects. Figure 5.10 shows the Simulink model used to simulate a unit-step response for the pole placement control system, using *saturation blocks* for the control input and the vertical speed of $\Delta\omega = \pm 50 rads^{-1}$ and $\Delta\dot{z} = \pm 1 ms^{-1}$, respectively. A block is included in the model that computes the continuous root mean square, RMS, of the

control effort to the plant. This RMS value is related to the amount of energy expended by the actuators, and serves as performance index for the various controllers.

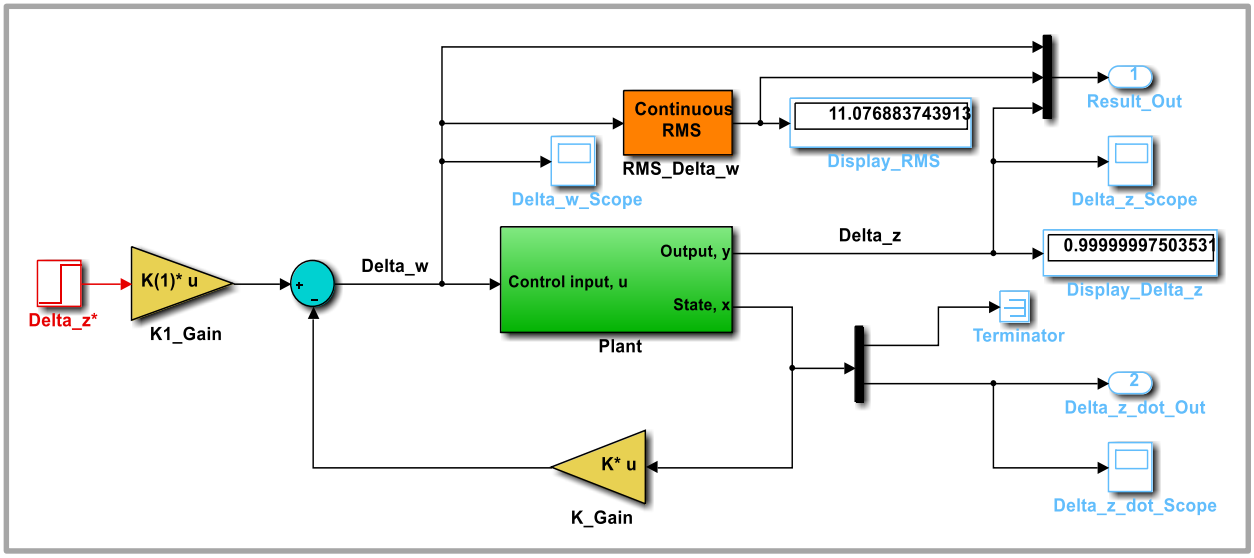


Figure 5.9. Simulink model setup for the full-state feedback closed-loop system.

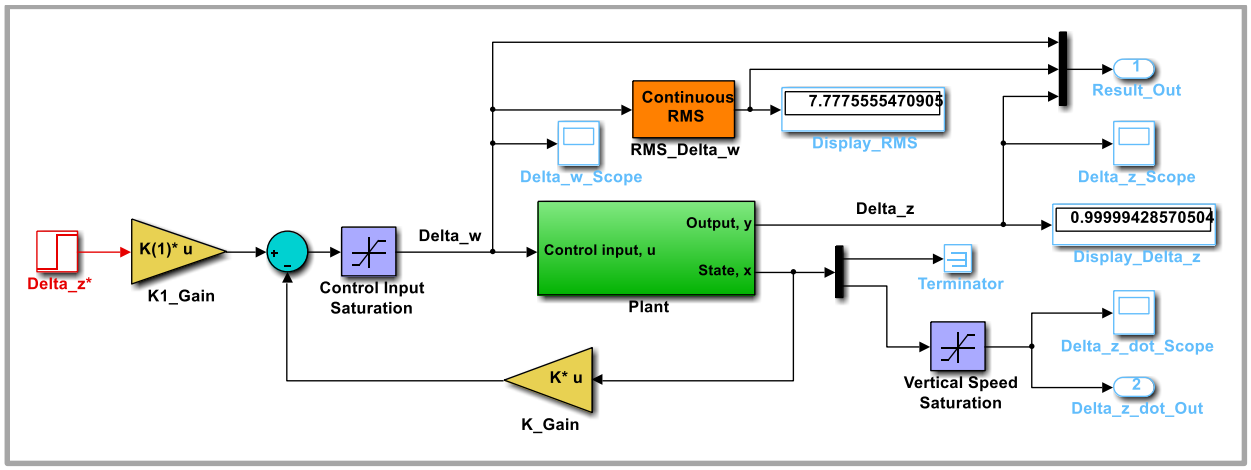


Figure 5.10. Full-state feedback closed-loop system model with saturation blocks.

5.2.1.3 Pole placement with saturation and time delay effects. Figure 5.11 shows the Simulink model used to simulate a unit-step response for the pole placement control system, with *transport delay block* used to implement the overall time delay, T_d, in the system.

5.2.1.4 PD-MRAC with and without nonlinear effects. Figure 5.12 shows the Simulink model used to simulate a unit-step response for the MRAC control system without PD controller and the *transport delay block*.

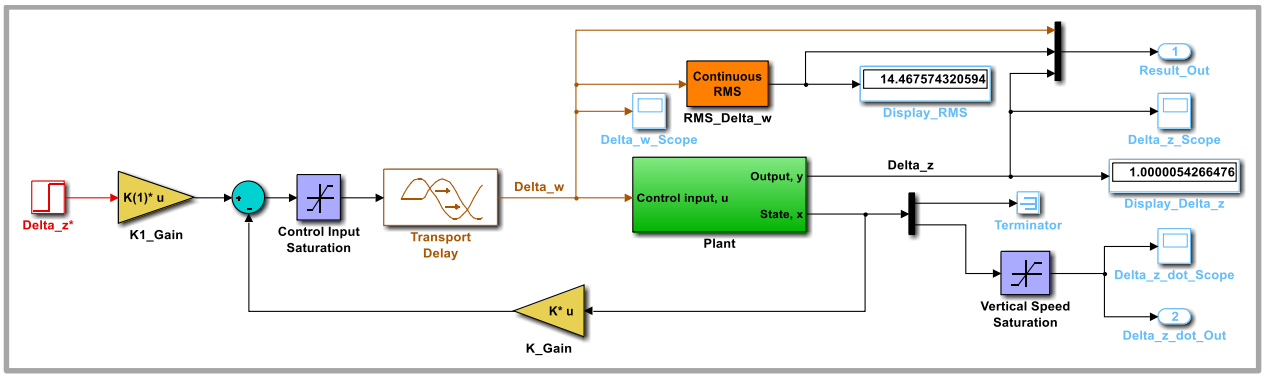


Figure 5.11. Full-state feedback closed-loop system model with transport delay block.

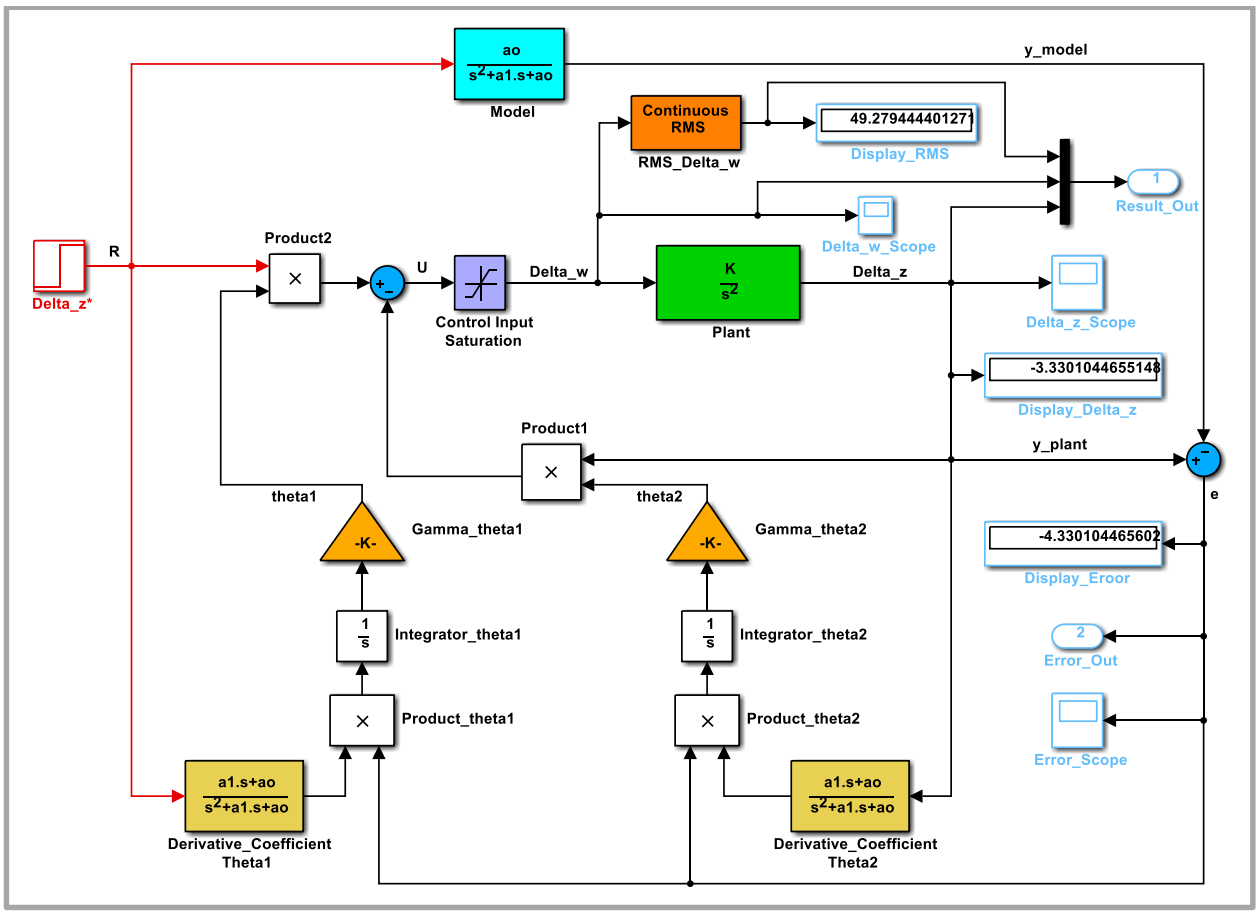


Figure 5.12. MRAC control system Simulink model without PD controller.

The MIT rule MRAC by itself does not guarantee convergence or stability [47], so a PD controller is included in the setup shown in Figure 5.12. Figure 5.13 shows the Simulink model used to simulate a unit-step response for the MRAC control system with the PD controller, which also include the *transport delay block*. The model also includes a suitable high pass filter (HPF), with damping ratio, $\xi_f = 1.0$ and natural frequency, $\omega_f = 100 rads^{-1}$.

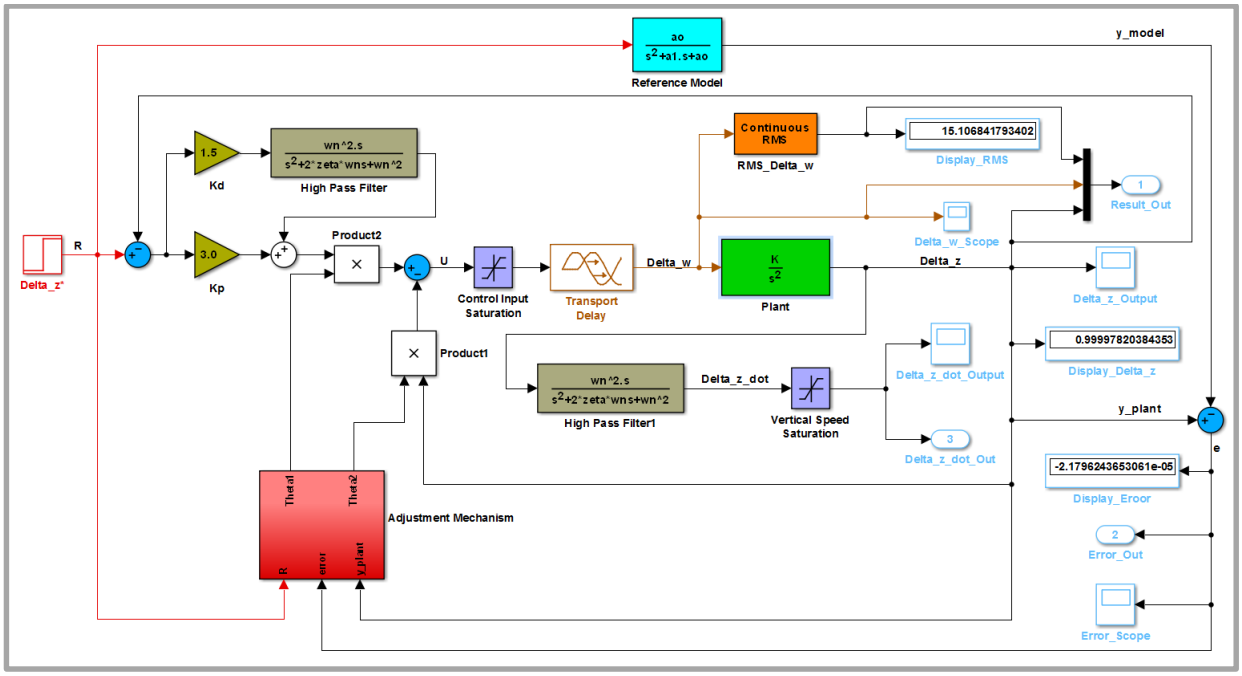


Figure 5.13. PD-MRAC control system Simulink model with transport delay block.

5.2.2 Experiments. This section presents experimental setup for implementation of the designed controllers on the AR.Drone 2.0. Figure 5.14 shows a general control setup for implementation of controllers. This setup was used for the controllers designed based on the first-order plant model (see *Section 3.2.5*) and the second-order plant model obtained from the black-box approach (see *Section 3.2.6*). The setup in Figure 5.15 was attempted for the implementation of the designed controllers based on the plant model obtained in (3.89) or (3.90). The first setup, Figure 5.14, cannot be used for the latter designed controllers, since the control input of the drone is the vertical linear speed, and that of the quadrotor model is the average angular speed.

The experiments were performed in an office environment with the drone's indoor hull attached. The drone was connected to its host PC using Wi-Fi, and data streaming, sending and receiving, was made possible using UDP (user datagram protocols), see Figure 5.16. UDP is a communications protocol, an alternative to TCP that offers a limited amount of service when messages are exchanged between computers in a network that uses IP.

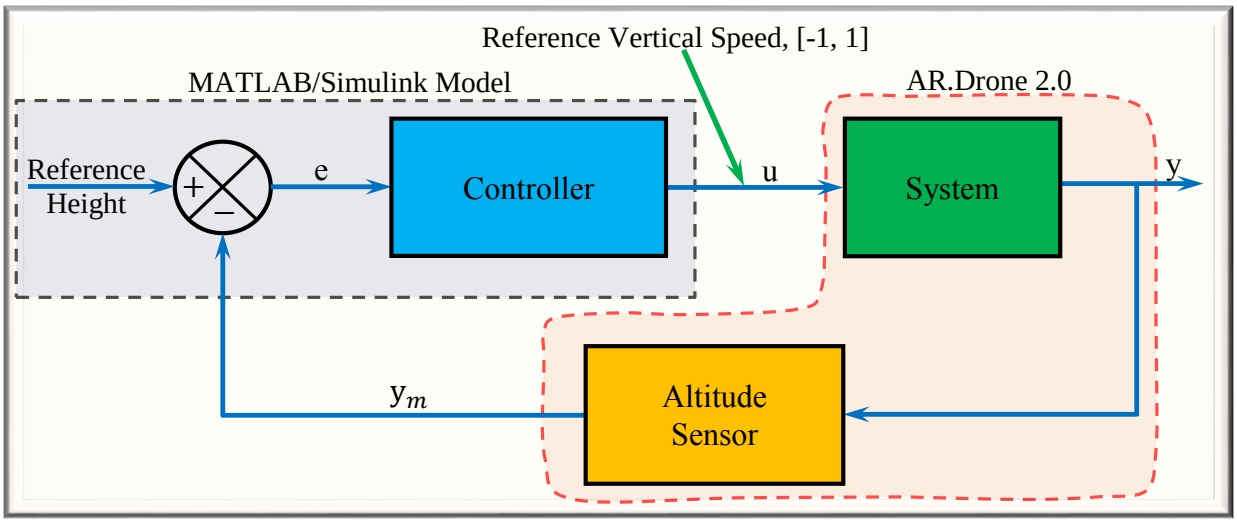


Figure 5.14. Generic control system for implementation.

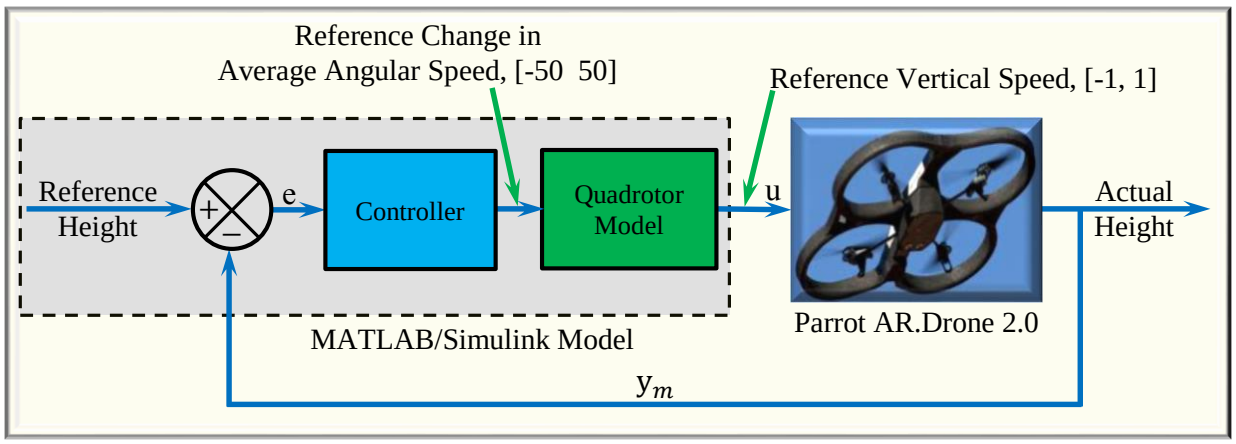


Figure 5.15. Intended control system for the implementation.

The drone has ultrasound sensor for ground altitude measurement (at the bottom). It has 1GHz 32 bit ARM Cortex A8 processor, 1GB DDR2 RAM at 200MHz, and USB 2.0 high speed

for extensions. The drone navigation data (from the sensors, cameras, battery, status, etc.) are received, and the control signals are sent, using AT commands. AT commands are combination of short text strings sent to the drone to control its actions.

The drone has 720p 30FPS HD front camera, 60 FPS vertical QVGA camera for ground speed measurement, 3 axis gyroscope, 3 axis accelerometer, 3 axis magnetometer for measuring the attitude angles. The drone has a 1000 mAh battery (3 cells) that can take up to about one and half hours to be fully charged, and can last for about fifteen minutes when the drone is in full operation. It has a dimension of 51.5cm (length) x 51.5cm (width) x 12cm (height), with a mass of 0.453kg for indoor operation, and has 2-pair of four rotors.

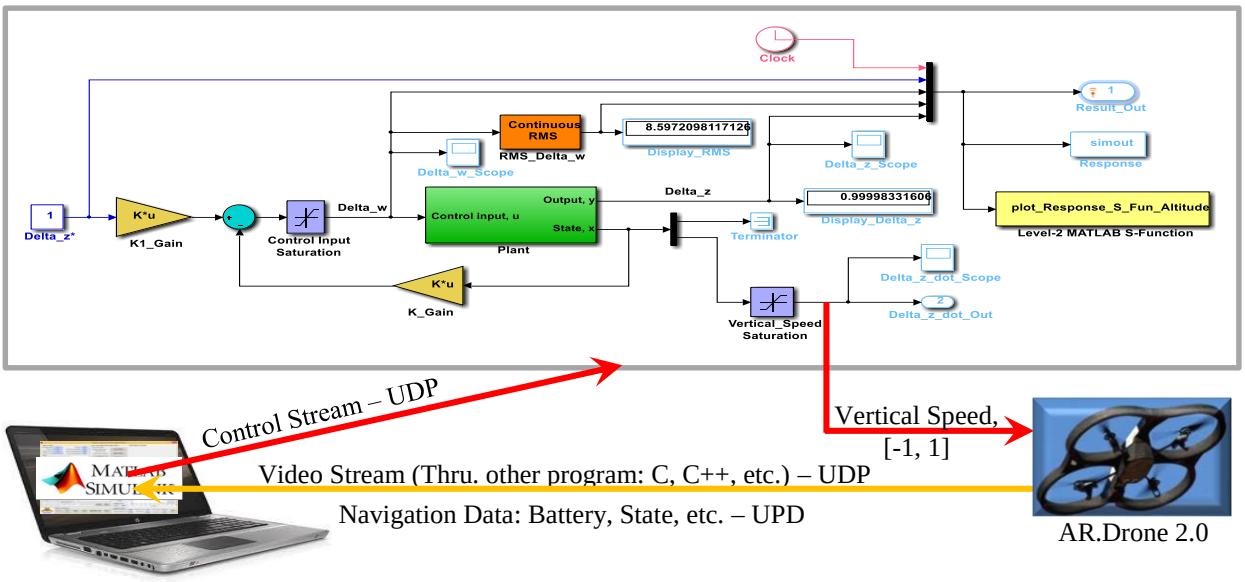


Figure 5.16. Experimental setup for the white-box approach model: full-state feedback.

Figure 5.16 shows an implementation setup using the full-state feedback control system, and Figures 5.17 and 5.18 show MATLAB Simulink and a robust GUI programs developed for the experiments. The GUI program was used to run the Simulink program, which contains the various developed altitude controllers. The GUI program works by first initializing the type of control mode, either to run the altitude designed controllers autonomously or manually control the

drone's different motions. This initialization step is conducted after the Wi-Fi connection has been established. Then, connection for data streaming from the drone is also established.

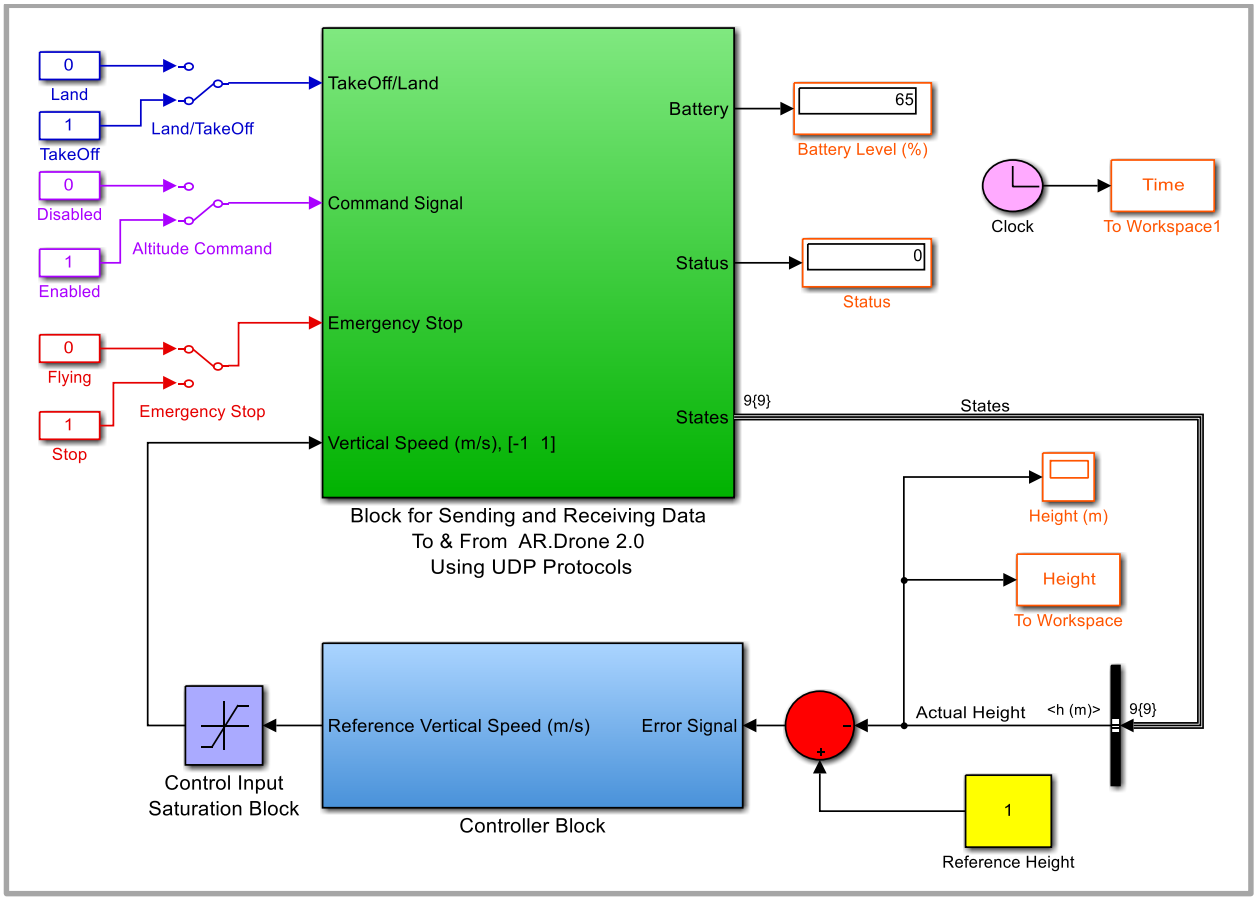


Figure 5.17. Experiment setup: Simulink program for controlling the AR.Drone 2.0.

The manual control mode allows the control of the various drone's motions, by selecting the type of motion and making the necessary settings. For example, one can select the yaw motion, and then set the direction of rotation and the angle of rotation. The manual control mode runs a MATLAB function containing codes for the various drone motions.

To autonomously control the drone's altitude motion using the designed controllers, one has to select the type of controller, then set the reference height. Multiple desired heights or waypoints can also be selected. The default parameter values of the controller selected can be used for the control or they can be changed. Then, the selected controller can be built for C/C++ code

generation, followed by establishing connection between the Simulink program and the drone, and finally real-time control signals sent to the drone for the altitude control. The interface also has an option for designing a new controller.

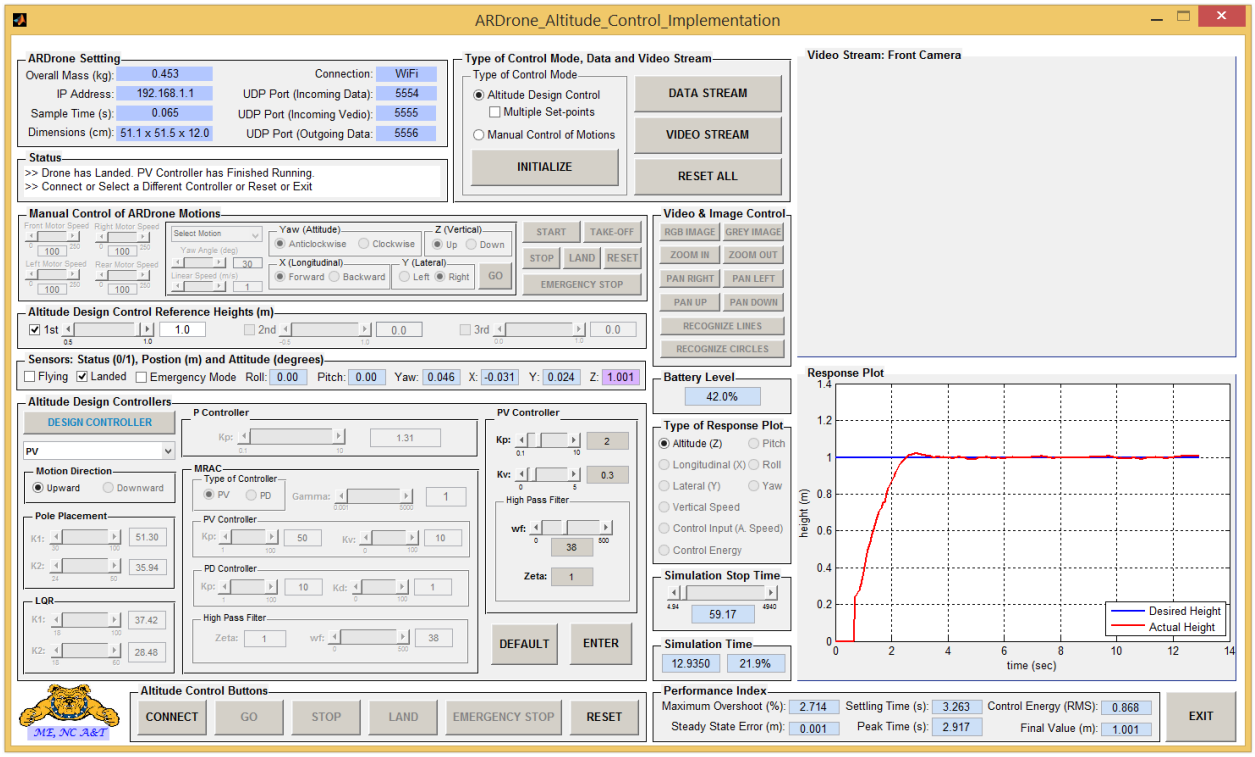


Figure 5.18. MATLAB GUI setup for the implementation.

The GUI interface has a section for displaying the drone's information such as its dimensions, type of connection, overall mass, and the control system sampling time. Also, it has sections that display, in real time, sensors information such as the position ($x, y,$ and z), attitude ($roll, pitch,$ and yaw), and the battery level. The Simulink control programs stopping times can also be set from the GUI. Real-time drone motion responses can be visualized. Performance index such as maximum overshoot, settling time, and control energy can also be displayed. Additionally, the interface has a section for video and image controls for future research work. The real-time response that displays in the GUI was achieved by using *S-function block* in the Simulink program. The real-time sensors information were also achieved by using MATLAB *Output block run-time*

object and an *event-listener* mechanism *add_exec_event_listener* command, and then synchronizing the *run-time object* with the Simulink execution.

5.3 Time-delay Estimation

The MATLAB/Simulink program setup developed for the simulations in estimating the time delay is shown in Figure 5.19. The vertical speed control input constraints, $[-1\ \ 1]ms^{-1}$, are applied using the *saturation block*. For the simulations, the time delay in the system is implemented using a *transport delay block*.

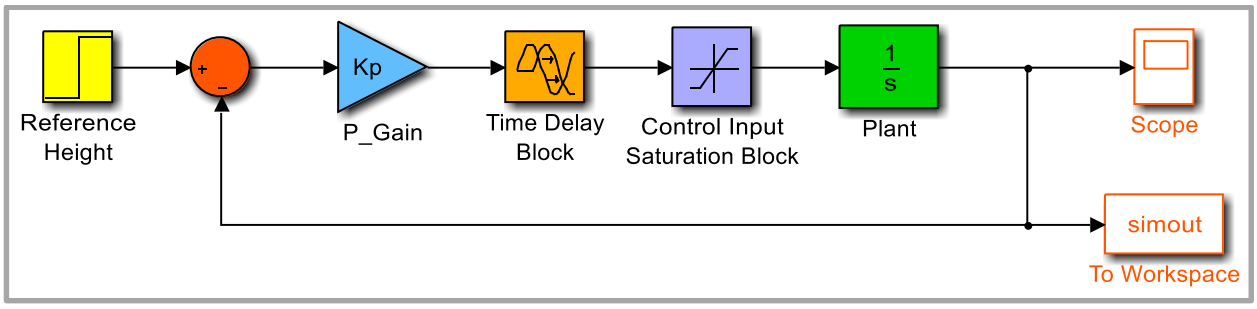


Figure 5.19. Simulink block diagram for P-feedback control system.

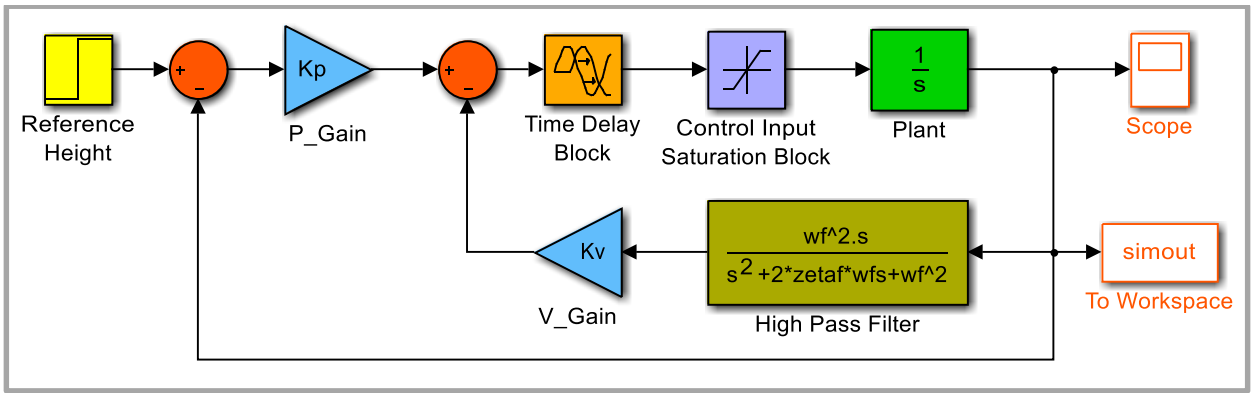


Figure 5.20. Simulink block diagram for PV-feedback control system.

5.4 Design of PV and PV-MRAC Controllers

Figure 5.20 shows the Simulink setup developed for conducting the PV control simulations, with the controller gains used in Figure 5.17 for the experiments. Suitable PV controller gains, K_p and K_v, are obtained to improve on the transient response performance. HPF

with damping ratio, $\xi_f = 1.0$ was used for the V controller. A suitable natural frequency, ω_f, value was designed for the filter.

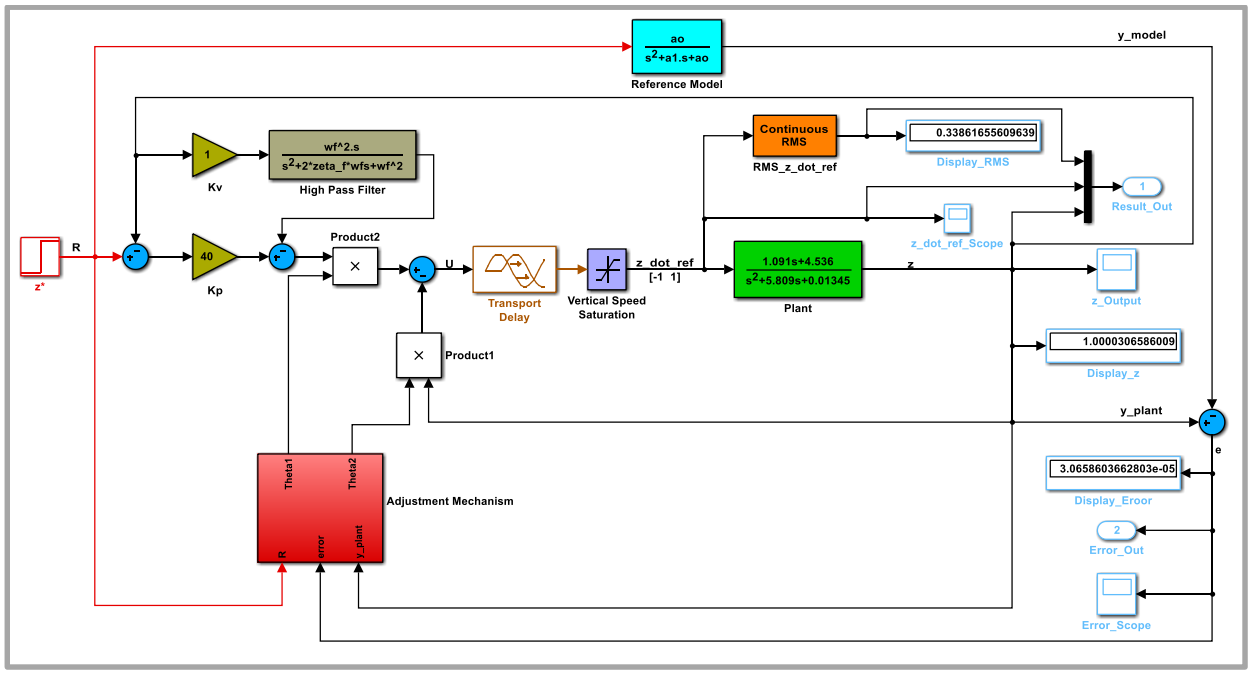


Figure 5.21. Simulink block diagram for PV-MRAC control system.

Figure 5.21 (see Figure 5.13 for a similar setup) shows the Simulink model used to simulate a unit-step response for the PV-MRAC control system with saturation and time delay effects. This setup uses the second-order plant model obtained from the black-box approach. The block model also include a suitable HPF, with damping ratio, $\xi_f = 1.0$ and natural frequency, $\omega_f = 38 rads^{-1}$.

5.5 Effects of System Uncertainty

5.5.1 Real stability radius computation. MATLAB programs were developed for the calculation of the real stability radius, and as stated in *Section 2.4.3*, MATLAB $fminbnd$ was used for the function minimization. The MATLAB/Simulink program setups developed for conducting the simulations of the P-feedback control systems are shown in Figures 5.22 and 5.23. $K_p = 1.31$, $T_d = 0.37s$, $\tau_n = 0.1304s$, and $K_{g_n} = 0.9451 rads^{-1}V^{-1}$ were used in the algorithms and simulations.

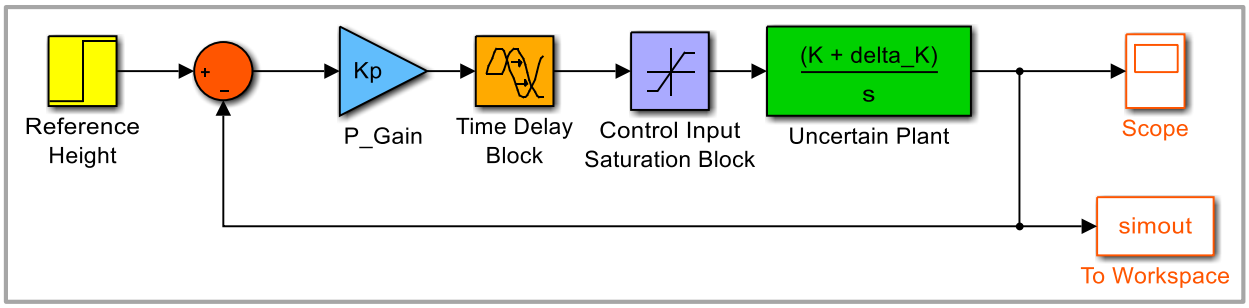


Figure 5.22. Simulink block diagram for P control system: uncertain first-order model.

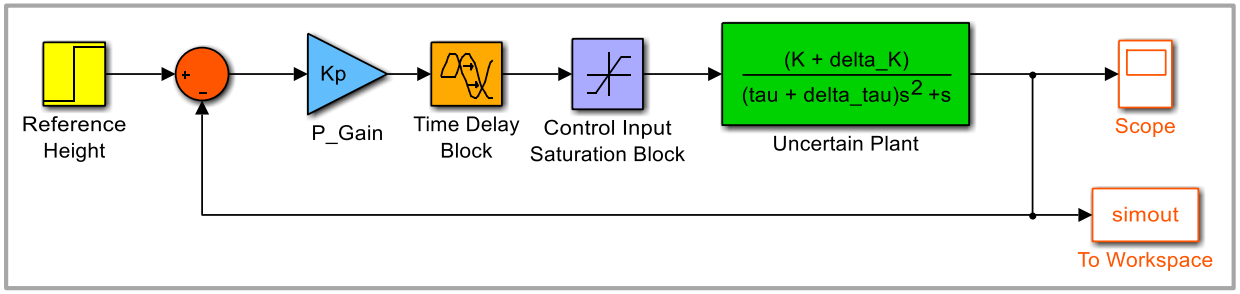


Figure 5.23. Simulink block diagram for P control system: uncertain second-order model.

5.5.1.1 First-order model. The real stability radius, $r_{\Re}$, of 0.8755 was obtained when $\omega^* = 2.8535 rads^{-1}$ and $\gamma^* = 0.0001$, thus the destabilizing perturbation matrix minimum norm, $\|\Delta\| = \|\Delta_B\| = 0.8755$. Therefore, $\Delta_{K_g} = \pm\|\Delta_B\|/K_P = \pm 0.6683 rads^{-1}V^{-1}$. Sample values of $\Delta_{K_g} = \pm 0.6683, \pm 0.4932, \pm 0.000$, and 1.000 were used to obtain simulation altitude responses.

5.5.1.2 Second-order model: reduced-form. Three situations were considered, depending whether the system is subjected to perturbations in **A** or **B** or both **A** and **B**.

5.5.1.2.1 Situation 1: $\Delta_A \neq 0$ and $\Delta_B = 0$. Here, $\mathbf{F}_B = 0$, and $\mathbf{F}_A = \mathbf{E} = \mathbf{I}_2$. $r_{\Re}$ of 0.7769 was obtained when $\omega^* = 17.3099 rads^{-1}$ and $\gamma^* = 0.9304$, thus $\|\Delta\| = \|[\Delta_B,\ \Delta_B]\| = \|\Delta_A\| = 0.7769$.

5.5.1.2.2 Situation 2: $\Delta_A = 0$ and $\Delta_B \neq 0$. Here, $\mathbf{F}_A = 0$, and $\mathbf{F}_B = \mathbf{E} = \mathbf{I}_2$. $r_{\Re}$ of 0.7769 was obtained when $\omega^* = 17.3099 rads^{-1}$ and $\gamma^* = 0.9415$, thus $\|\Delta\| = \|[\Delta_B,\ \Delta_B]\| = \|\Delta_B\| = 0.7769$.

5.5.1.2.3 Situation 3: $\Delta_A \neq 0$ and $\Delta_B \neq 0$. Here, $\mathbf{F}_A = \mathbf{F}_B = \mathbf{E} = \mathbf{I}_2$. $r_{\Re}$ of 0.5494 was obtained when $\omega^* = 17.3918 rads^{-1}$ and $\gamma^* = 0.9421$, thus the destabilizing perturbation matrix minimum norm, $\|\Delta\| = \|[\Delta_B,\ \Delta_B]\| = 0.5494$.

Table 5.1

Possible Destabilizing Perturbation Matrices Minimum Norm, $\|\Delta\|$: Experiments

	Case 1	Case 2	Case 3
	$\Delta_{K_g} = 0.0100$ & $\Delta_\tau = 0.0000$	$\Delta_{K_g} = 0.0100$ & $\Delta_\tau = -0.0010$	$\Delta_{K_g} = -0.2000$ & $\Delta_\tau = -0.0100$
$\|\Delta\| = \|[\Delta_A, \Delta_B]\|$	0.1010	0.1847	1.5272

As it can be observed, *Situation 3* gives the smallest of $\|\Delta\| = 0.5494$. Now, from the sample of possible values of Δ_A and Δ_B shown in Table 3.1 (see *Section 3.2.7.2*), the possible values of $\|\Delta\|$ computed are shown in Table 5.1. Sample pair values of $(\Delta_{K_g}, \Delta_\tau)$ for *Case* 1, *Case* 2, *Case* 3, $(0.0000, 0.0000)$, and $(1.0000, 0.5000)$ were used to obtain simulation altitude responses.

5.6 Autonomous Waypoints Tracking and Effects of Disturbance Rejection

5.6.1 Autonomous waypoints tracking. The designed PV and PV-MRAC control systems were used to track multiple specified waypoints autonomously. For example, the drone was made to move to a desired height of $1m$, then $2m$, and then back to $1m$. The waiting time was $5s$ at each of these heights. The drone finally moved to height of $1.5m$ for a period of about $10 - 15s$ before landing.

5.6.2 Effects of disturbance rejection: quadrotor payload. The robustness of the PV-MRAC controller was tested against a baseline, PV controller, using the payload capability of the

drone and the introduction of disturbances to the system. To accomplish this, three cases were considered.

5.6.2.1 Case 1. Masking tape was used to attach a mass at the top of the drone, see Figure 5.24 for an example. The controllers were tested using different masses attached to the top of the drone. The setup was used to determine the stability bounds within which the changing mass-inertia parameters of the system due to the acquired object will not destabilize the drone.

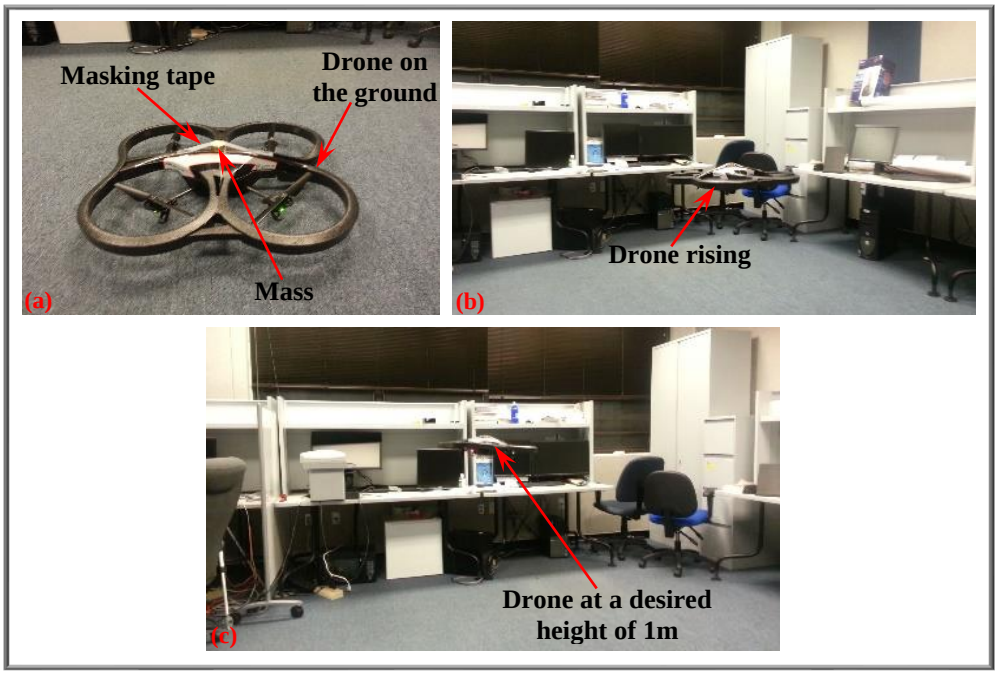


Figure 5.24. Experimental setup: mass attached at the top.

5.6.2.2 Case 2. Here, a $100g$ mass was attached to the end of an approximately $45cm$ length of rope, which was hooked to the drone, see Figure 5.25. Masking tape was used to hold firm the rope across the top and the bottom of the drone. The sensors, which include the one for altitude readings, at the bottom of the drone were not covered. In this second case, apart from the drone carrying an extra load, the swinging mass-rope created an oscillating (pendulum) disturbances to the system.

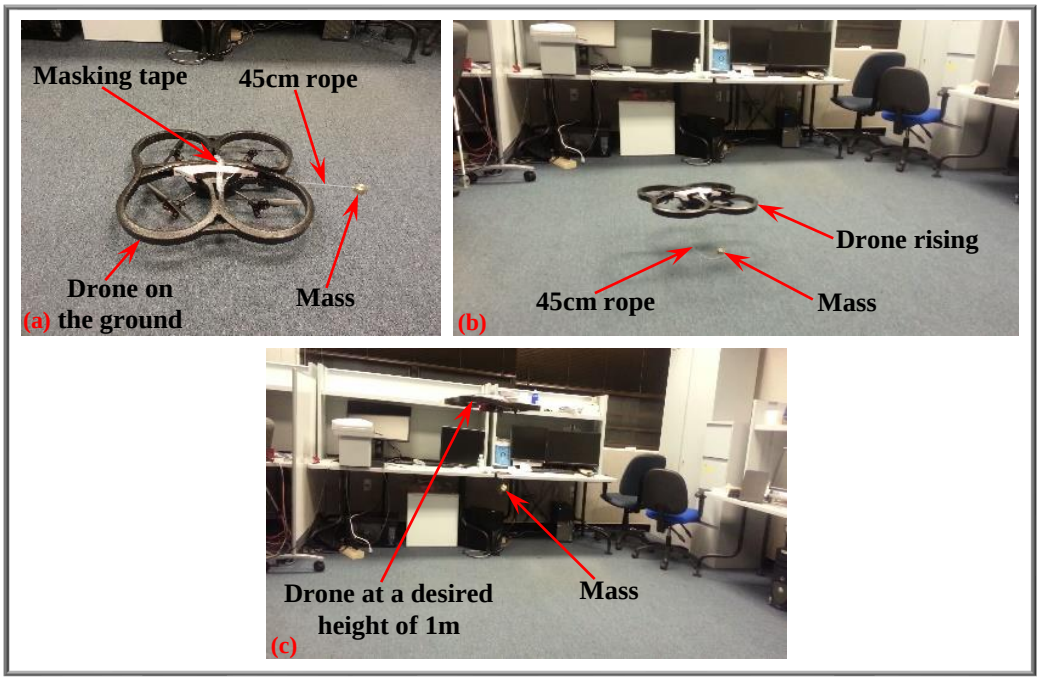


Figure 5.25. Experimental setup: $100g$ mass hanging by rope.

5.6.2.3 Case 3. In this case, the designed controllers were used to demonstrate the stability behavior of the drone undergoing a range of instantaneous step payload changes. Here, the masses were attached to a sticking masking tape at the top of the drone, after it has stabilized to a desired height of $1m$, see Figure 5.26.

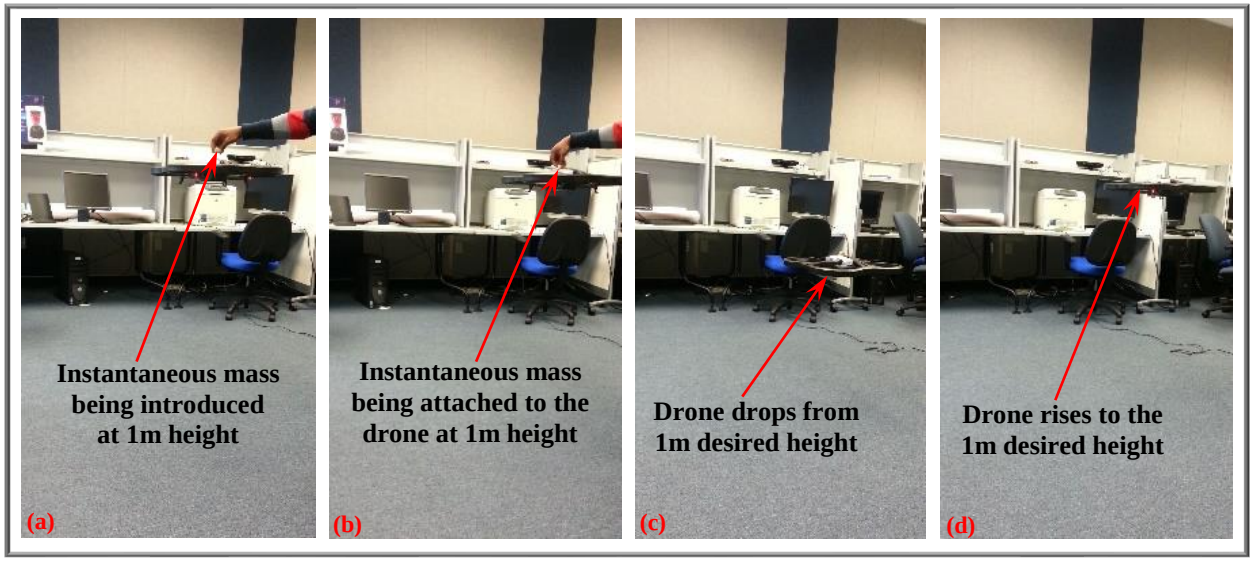


Figure 5.26. Experimental setup: instantaneous step mass.

CHAPTER 6

Results and Discussion: Ground Robot

This chapter presents the results obtained via simulations and experiments along with discussions regarding the ground robot.

6.1 Individual Behaviors Control Algorithms

The time domain Simulink simulations were conducted over a 10-second duration for each model (refer to the simulation setups in Figures 4.2 to 4.5). The trajectories in Figure 6.1 were obtained by using proportional gains of 0.5 and 4.0 for K_v and K_h respectively. The final goal point was $(4.9920m, 5.0036m)$, compared to the desired goal of $(5m, 5m)$ for the (5m, 9m, π) initial state. Trajectories in Figure 6.2 were obtained using $K_\rho = 3.0$, $K_\alpha = 8.0$ and $K_\beta = -3.0$; the final pose was $(4.9451m, 5.0004m, 2.7693rad)$, compared to the desired pose of $(5m, 5m, \pi)$ for the (5m, 9m, π) initial state.

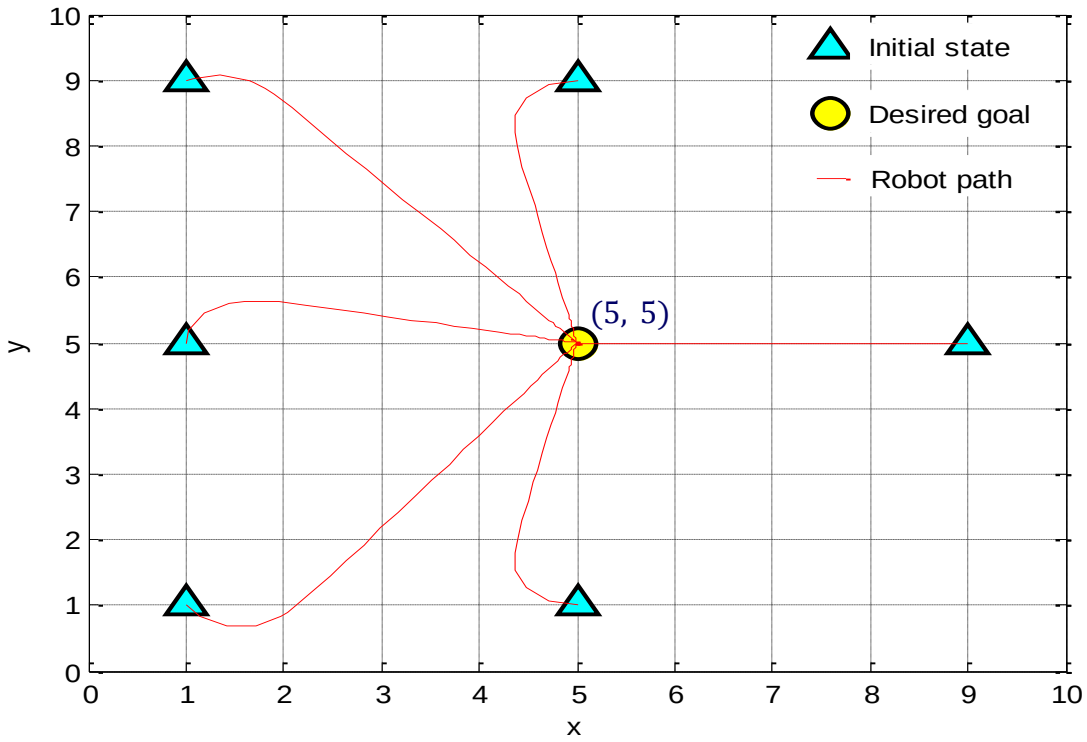


Figure 6.1. Move to a point: simulation.

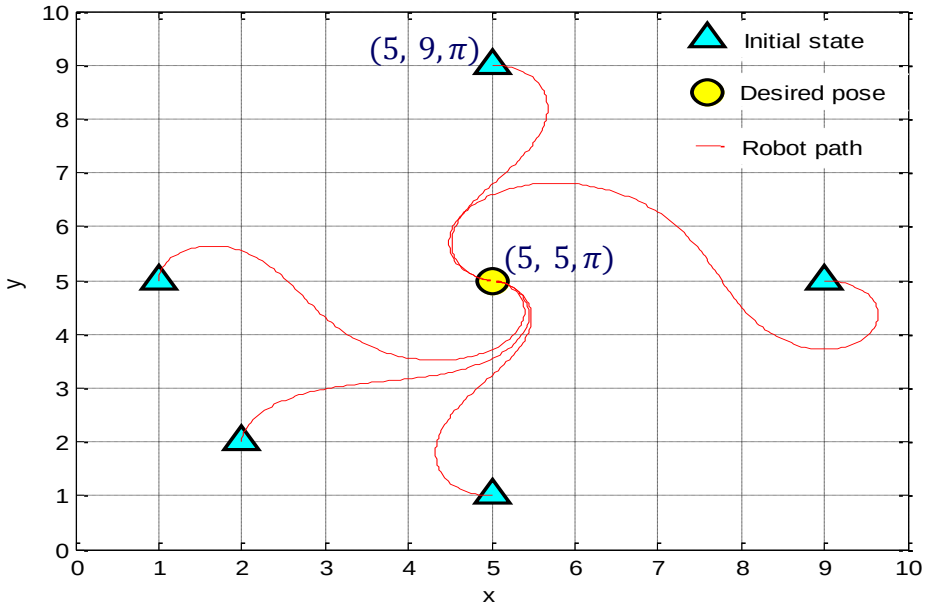


Figure 6.2. Move to a pose: simulation.

The trajectories in Figure 6.3 were obtained with $K_d = 0.5$ and $K_h = 1.0$. The robot was driven at a constant speed of $v = 1.0ms^{-1}$. The trajectory in Figure 6.4 was obtained using $K_h = 5$, PID controller gains of $K_p = 1.0, K_I = 0.5,$ and $K_D = 0.0$, and the goal was a point generated to move around the unit circle with a sampling frequency of $0.2Hz$.

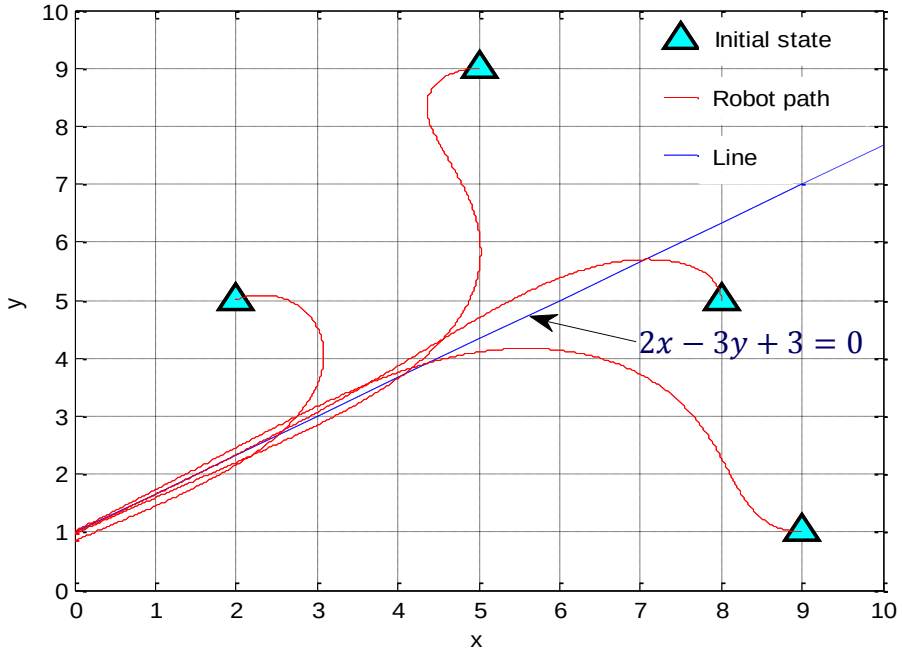


Figure 6.3. Follow a line: simulation.

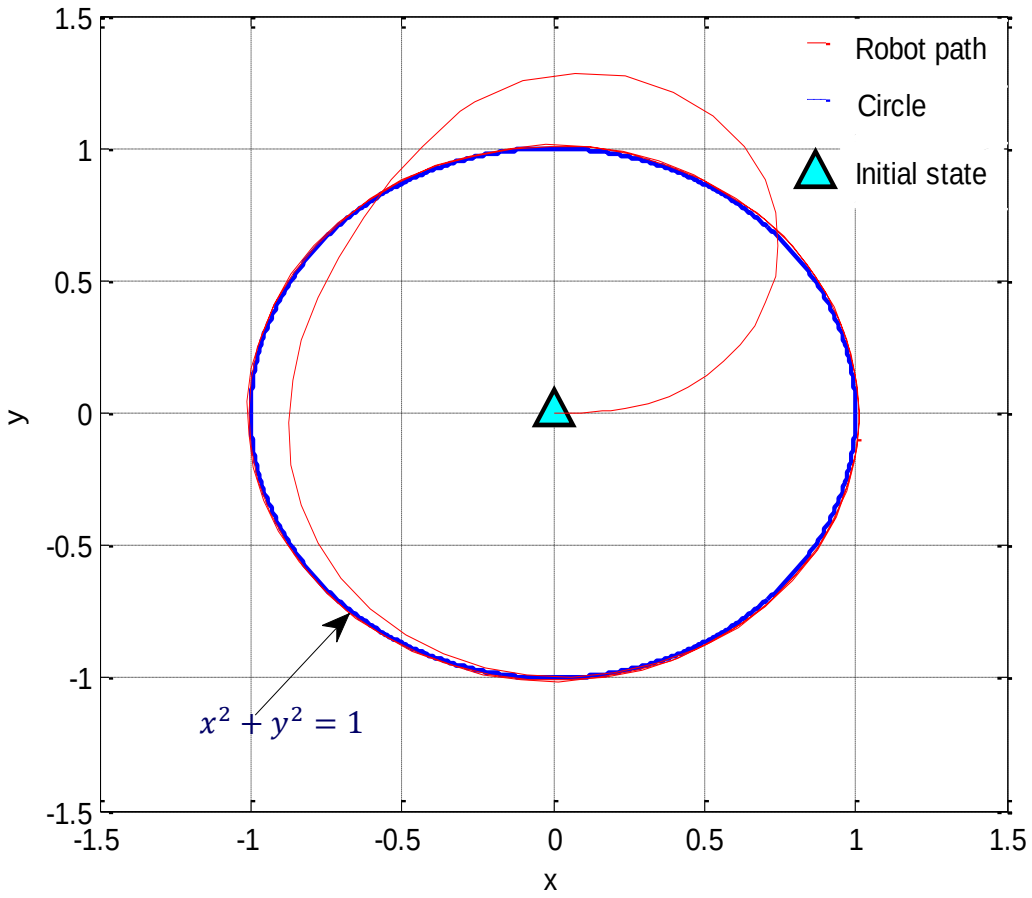


Figure 6.4. Follow a circle: simulation.

6.2 Navigation System Control Algorithm

6.2.1 Simulations. The trajectory shown in Figure 6.5 (refer to the simulation setup in Figure 4.6) was obtained by using PID controller gains of $K_p = 5.0, K_I = 0.01$ and $K_D = 0.1, \alpha = 0.6, \epsilon = 0.05, v = 0.1ms^{-1},$ initial state of $(0m, 0m, 0rad)$, and desired goal of $(1m, 1m, \pi/2)$. The final goal point associated with the simulation was $(1.0077m, 0.9677m, 1.6051rad)$, and the average stabilization time was about 35s.

6.2.2 Experiments. The trajectory shown in Figure 6.6 (refer to a similar experimental setup in Figure 4.12) was obtained by using PID controller gains of $K_p = 1000, K_I = 1000,$ and $K_D = 5$ for the position control and $K_p = 10, K_I = 0,$ and $K_D = 1$ for the velocity control, $\epsilon = 0.01,$ $v = 0.5ms^{-1},$ initial state of $(0m, 0m, 0rad)$, and desired pose of

$(2m, 0m, 0rad)$. The final pose associated with the experiment was $(2.0197m, 0.0266m, -0.0096rad)$, and the average stabilization time was about $44s$.

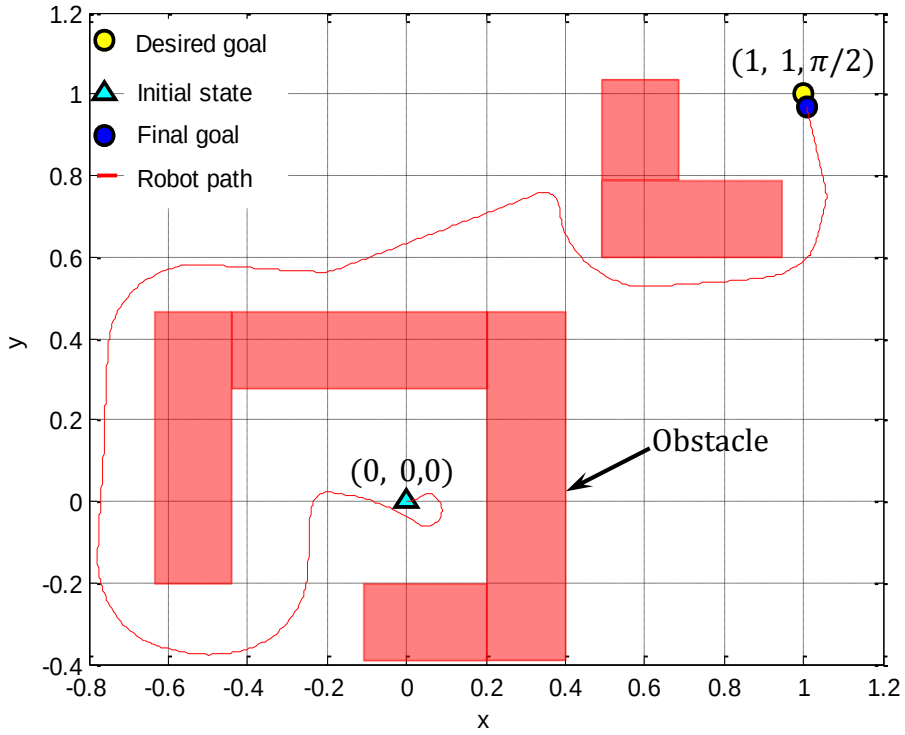


Figure 6.5. Trajectory in xy-plane: simulation.

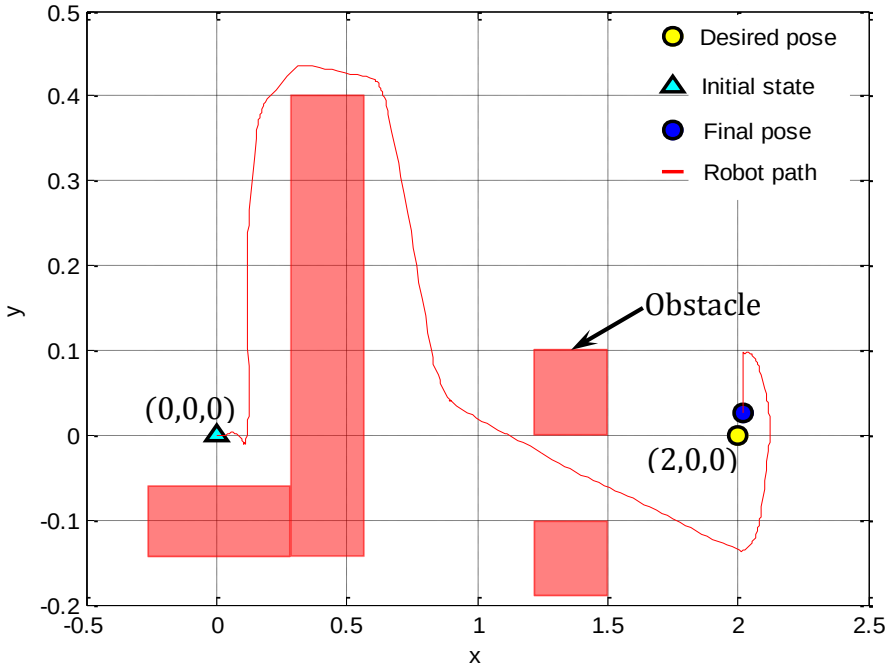


Figure 6.6. Trajectory in xy-plane: experiment.

Although there were steady-state errors in the values obtained, the results were encouraging. Possible causes of the errors could include: Firstly, due to the friction between the robot wheels and the floor. Secondly, imperfection in the sensors. Thirdly, unmodeled factors (e.g., friction and backlash) in the mechanical parts of the DC motor. Moreover, despite the apparent simplicity of the kinematic model of a WMR, the existence of nonholonomic constraints (due to state or input limitations) turns the PID-feedback stabilizing control laws into a considerable challenge. According to Brockett's conditions, a continuously differentiable, time-invariant stabilizing feedback control law cannot be obtained [77].

Note that during the experiments the robot sometimes got lost or wandered around before arriving at the desired pose. This is because the navigation system is not robust. It was built using a low-level planning based on a simple model of a point mass and the application of a linear PID controller.

CHAPTER 7

Results and Discussion: Aerial Robot

This chapter presents the results obtained via analytical, simulations, and experiments along with discussions regarding the aerial robot.

7.1 Control of Quadrotor Motions: White-box Approach (Nonlinear Model)

All the time domain Simulink simulations were carried out under 10-second duration for each model (refer to the simulation setups in Figures 5.2 to 5.6). The responses, using the various control algorithms, are shown in Figures 7.1 to 7.3.

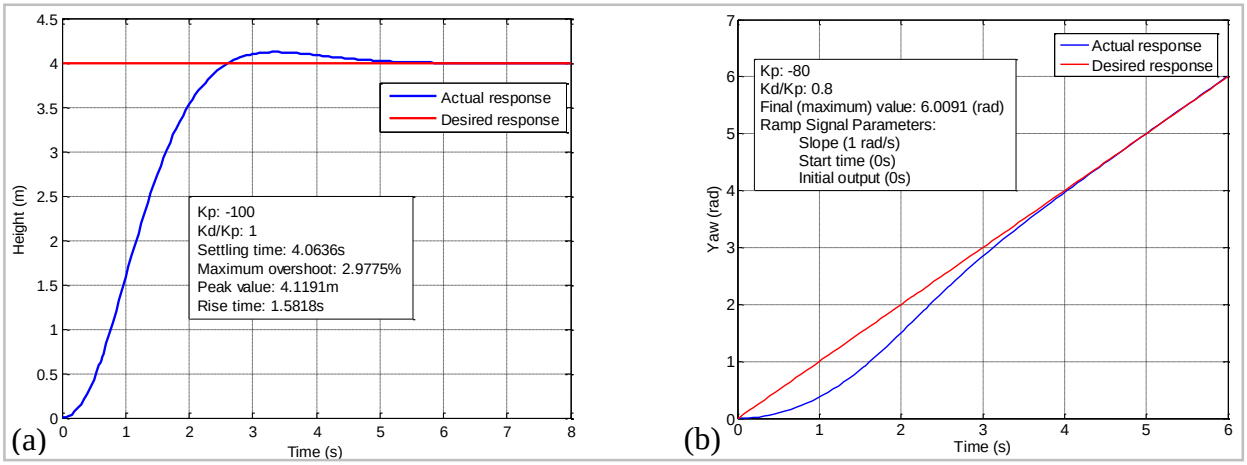


Figure 7.1. (a) Altitude control: step response (b) yaw control: ramp response.

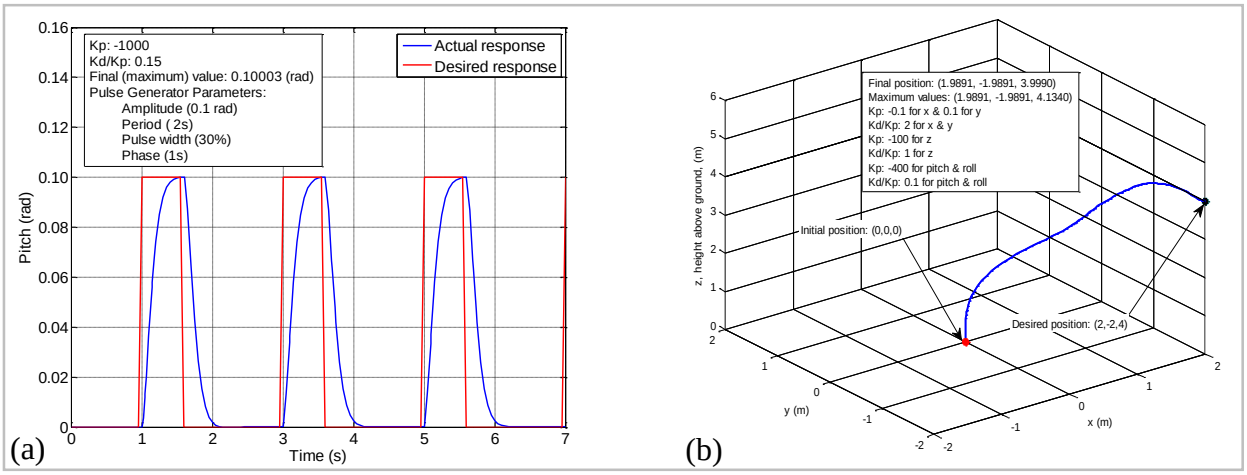


Figure 7.2. (a) Pitch control: pulse response (b) position control: trajectory.

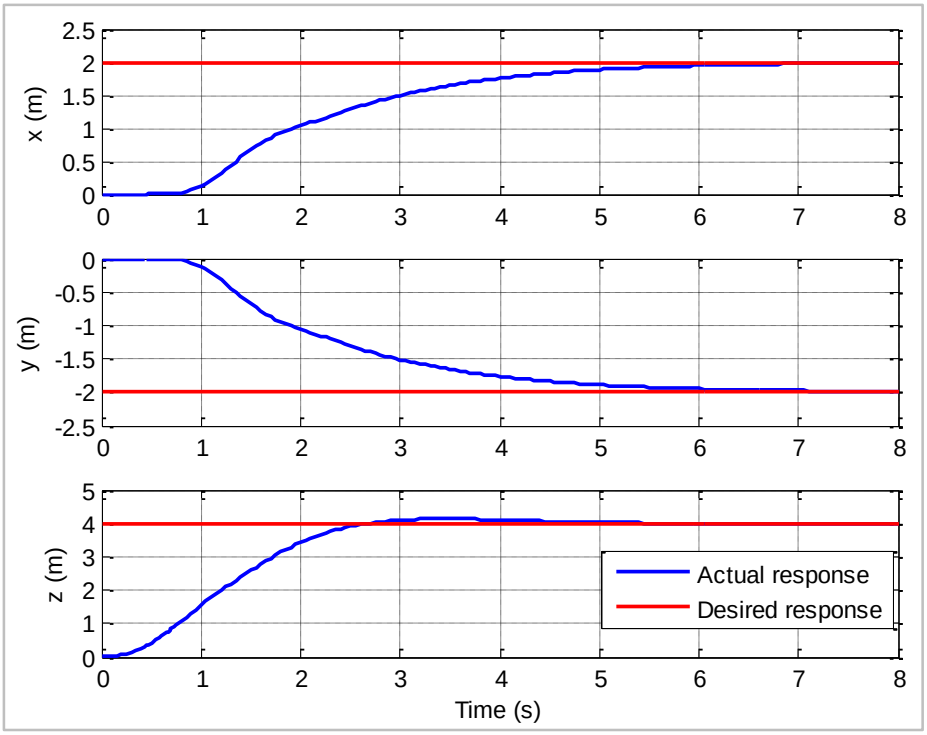


Figure 7.3. Position control: step response.

The PD-gains in this research were obtained by tuning to obtain the satisfactory responses. The various PD-gains, some performance indicators, and the input signals used to obtain the various responses are depicted in the figures. Even though there were steady-state errors, the results were encouraging. One of the possible causes of the errors in the steady state is in the modeling of the quadrotor's dynamics and the kinematics such as unmodeled aerodynamic effects and uncertainties (e.g., wind disturbance and saturation). In addition, according to Brockett's conditions, feedback control laws, which are a continuously differentiable, time-invariant, cannot be used to obtain error-free stabilization [77].

7.1.1 Effects of PD-gains. Table 7.1 and Figure 7.4a show how the range of values of the P-gains, keeping the D-gain at a constant value, affects the altitude responses during 8-second simulations. Theoretically, the trend in the values conform. For example, as the K_p increases at constant K_d, the maximum overshoot increases and the rise time decreases [44].

Table 7.1

Effect of the P-Gains on the Altitude Response: Desired Height, $z^* = 4m$

K_p	K_d/K_p	Maximum Overshoot (%)	Settling Time (s)	Rise Time (s)	Peak Value (m)	Peak Time (s)	Absolute Steady-State Error (m)
-200.0	1.0	0.0000	3.3914	1.8278	3.9999	8.0000	0.000086
-150.0	1.0	0.0724	2.9407	1.7048	4.0029	4.5500	0.000007
-100.0	1.0	2.9775	4.0636	1.5818	4.1191	3.3500	0.000751
-80.0	1.0	6.0702	4.6391	1.5520	4.2428	3.3000	0.003100

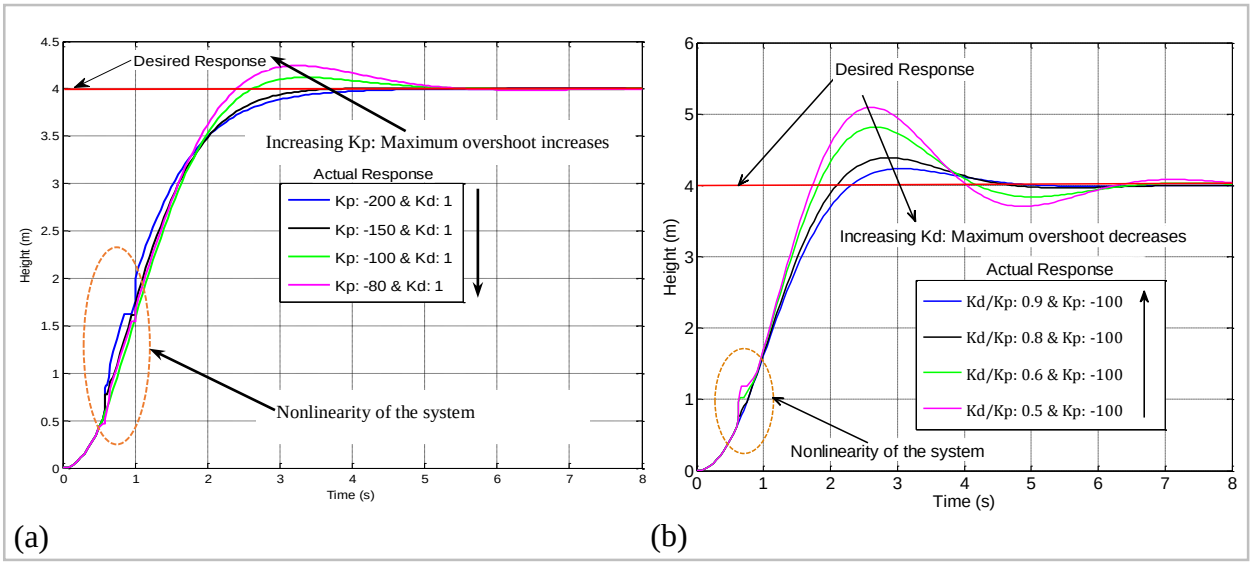


Figure 7.4. Effects of PD-gains (a) varying P-gains (b) varying D-gains: step response.

Table 7.2 and Figure 7.4b show how the range of values of the D-gains, keeping the P-gain at a constant value, affect the altitude responses during 8-second simulations. Theoretically, the trend in the values conform. For example, as the K_d increases at constant K_p, the maximum

overshoot decreases and there are minor changes in the rise time [44]. The non-smooth nature of the responses are as a result of the nonlinearity in the system dynamics and kinematics.

Table 7.2

Effect of the D-Gains on the Altitude Response: Desired Height, $z^* = 4m$

K_p	K_d/K_p	Maximum Overshoot (%)	Settling Time (s)	Rise Time (s)	Peak Value (m)	Peak Time (s)	Absolute Steady-State Error (m)
−100.0	0.9	5.8473	4.2467	1.4405	4.2339	3.0500	0.000016
−100.0	0.8	9.6776	4.2262	1.3252	4.3871	2.8500	0.003200
−100.0	0.6	20.3730	5.8530	1.1580	4.8149	2.6500	0.021600
−100.0	0.5	27.3765	7.2984	1.0993	5.0951	2.6000	0.036100

7.1.2 Challenges and contribution. The main challenges were getting the real-time parameters from the Simulink models and plotting of state responses to display in the GUI. As mentioned in *Section 5.1.2*, these were achieved using the MATLAB *block run-time object* and *S-function block*, respectively. The latter approach comes with challenges in implementation for real-time application, in C/C^{++} code generation.

This section of research has contributed the following: firstly, research and put together the theories of the dynamics and kinematics, and PD-feedback controllers for position and attitude of quadrotor UAVs. Secondly, it used MATLAB/Simulink models, developed by using similar models presented in [39], to design and validate the control algorithms. Furthermore, the research has demonstrated how MATLAB GUI can be used to run the Simulink models, exchanging real-time data between the GUI and the Simulink models.

7.2 Control of Quadrotor Altitude Motion: White-box Approach (Linearized Model)

This section presents the simulation and experimental results for the autonomous altitude control of the quadrotor. The results presented here are for the controllers designed based on the linearized model obtained from the quadrotor's nonlinear dynamics and kinematics.

7.2.1 Full-state feedback without nonlinear effects: simulations. Tables 7.3 and 7.4 shows the transient response properties, the controller gains, and the control RMS values of the pole placement controller by varying the real and imaginary parts of the desired closed-loop poles, respectively. Figures 7.5a and 7.5b, respectively, show the corresponding plots of the altitude responses.

Table 7.3

Pole Placement Altitude Responses without the Nonlinear Effects: Varying β_1

I	K	M_o (%)	t_s (s)	t_p (s)	E_{ss} (m)	Control Energy (RMS Value)
$-1.5 \pm 1.2i$	[30.14 24.51]	1.97	1.88	2.60	-3.04×10^{-4}	5.54
$-1.8 \pm 1.2i$	[38.23 29.41]	0.90	1.86	2.60	-5.72×10^{-5}	6.41
$-2.2 \pm 1.2i$	[51.30 35.94]	0.31	1.78	2.60	-5.72×10^{-6}	7.78
$-2.6 \pm 1.2i$	[66.99 42.48]	0.11	1.67	2.60	-4.86×10^{-7}	9.35
$-3.0 \pm 1.2i$	[85.28 49.01]	0.04	1.55	2.60	-2.50×10^{-8}	11.08

The results in the tables show that as the real part increases (i.e., moving the poles to the left in the complex plane) the gains values increase, M_o and t_s decreases at a constant t_p, and the RMS value increases. Also, as the imaginary part increases (i.e., moving the poles to the upwards) the gains values increase, M_o increases, t_s and t_p decreases, and the RMS value increases.

Table 7.4

Pole Placement Altitude Responses without the Nonlinear Effects: Varying β_2

I	K	M_o (%)	t_s (s)	t_p (s)	E_{ss} (m)	Control Energy (RMS Value)
$-2.6 \pm 0.1i$	[55.30 42.48]	0.00	2.24	4.94	-3.49×10^{-5}	7.72
$-2.6 \pm 0.8i$	[60.45 42.48]	0.00	1.96	3.90	8.04×10^{-6}	8.43
$-2.6 \pm 1.2i$	[66.99 42.48]	0.11	1.67	2.60	-4.86×10^{-7}	9.35
$-2.6 \pm 1.6i$	[76.14 42.48]	0.61	1.38	1.95	-4.16×10^{-6}	10.62
$-2.6 \pm 2.0i$	[87.90 42.48]	1.68	1.13	1.56	3.88×10^{-6}	12.26

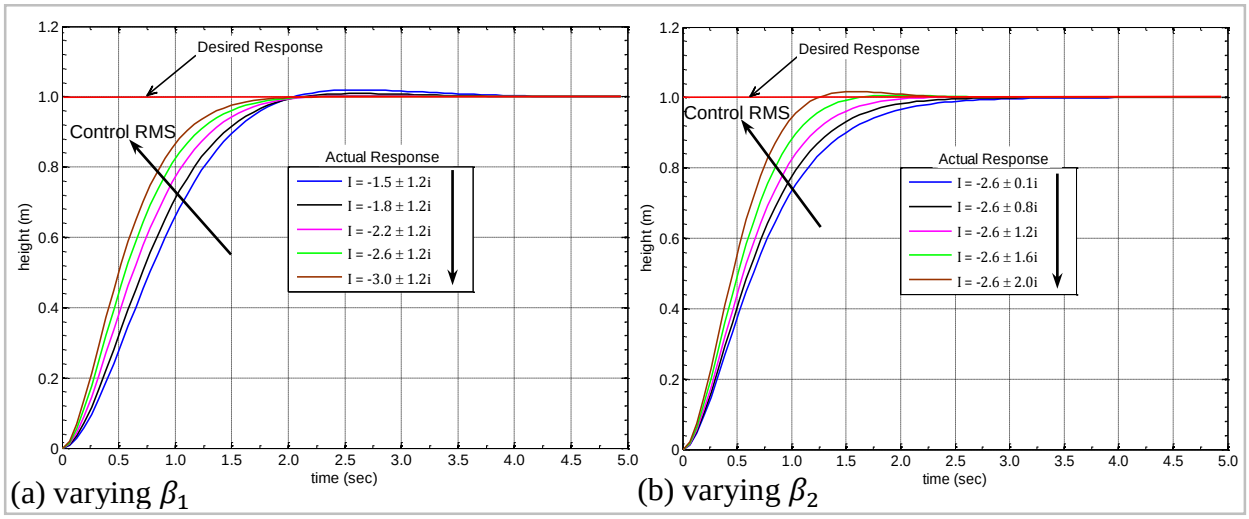


Figure 7.5. Pole placement altitude responses without the nonlinear effects.

Figures 7.6a and 7.6b shows the pole placement control input and vertical speed plots by varying the real part of the desired poles. The results show that as the real part increases, the RMS value increases. Also, looking at both plots, average angular speed, control input, of $\Delta\omega_a = \pm 50 rads^{-1}$ satisfies the drone's saturation vertical speed of $\Delta z_{ref} = \pm 1 ms^{-1}$.

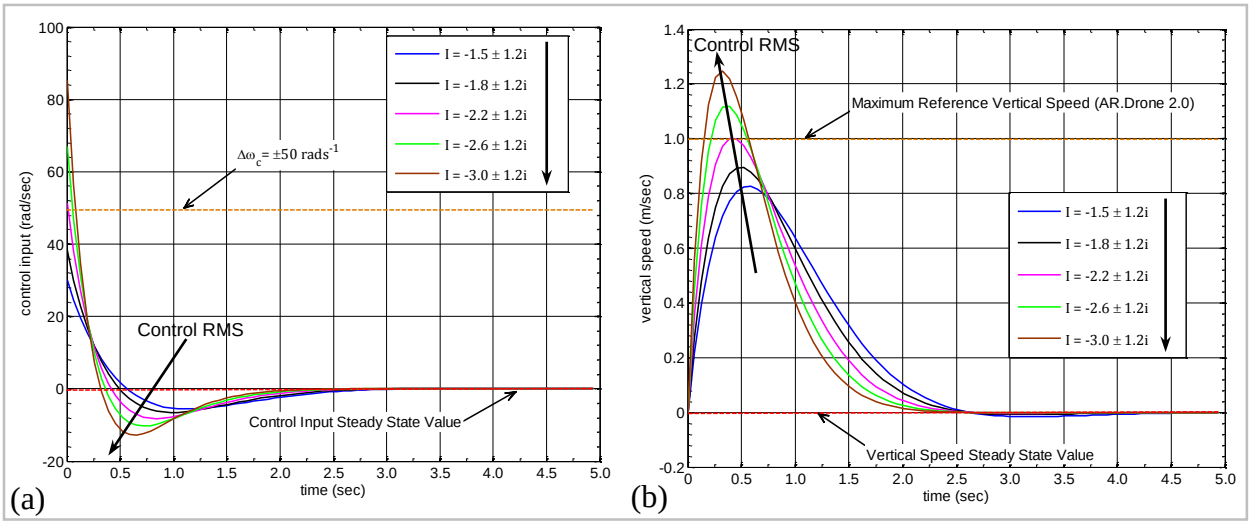


Figure 7.6. Pole placement without the nonlinear effects: (a) control input (b) vertical speed.

Table 7.5

LQR Controller Altitude Responses without the Nonlinear Effects: Varying Q

μ	K	M_o (%)	t_s (s)	t_p (s)	E_{ss} (m)	Control Energy (RMS Value)
50	[18.71 18.86]	2.46	3.70	3.19	0.004	3.92
100	[26.46 23.07]	1.87	2.02	2.80	1.55×10^{-4}	5.01
200	[37.42 28.48]	1.21	1.81	2.54	-1.41×10^{-4}	6.37
300	[45.83 32.38]	0.82	1.72	2.41	-6.59×10^{-5}	7.32
450	[56.12 36.97]	0.46	1.65	2.41	-1.95×10^{-5}	8.39

Tables 7.5 and 7.6 show the transient response properties, the controller gains, and the RMS values of the LQR controller by varying Q and R, respectively. Figures 7.7a and 7.7b show the corresponding plots of the responses, respectively. The results show that as the weighting parameter, μ, of Q increases, the gains values increase, M_o, t_s, and t_p decreases, and the RMS

value increases. Also, as the weighting parameter, μ, of R increases, the gains values decrease, M_o, t_s, and t_p increases, and the RMS value decreases.

Table 7.6

LQR Controller Altitude Responses without the Nonlinear Effects: Varying R

μ	K	M_o (%)	t_s (s)	t_p (s)	E_{ss} (m)	Control Energy (RMS Value)
0.1	[118.32 62.71]	0.00	1.56	4.94	-2.97×10^{-7}	13.59
0.5	[52.92 35.56]	0.56	1.67	2.41	-2.90×10^{-5}	8.07
1.0	[37.42 28.48]	1.21	1.81	2.54	-1.41×10^{-4}	6.37
1.5	[30.55 25.15]	1.61	1.93	2.67	-1.23×10^{-4}	5.54
2.0	[26.46 23.07]	1.87	2.02	2.80	1.55×10^{-4}	5.01
2.5	[23.66 21.60]	2.07	3.10	2.93	7.00×10^{-4}	4.63

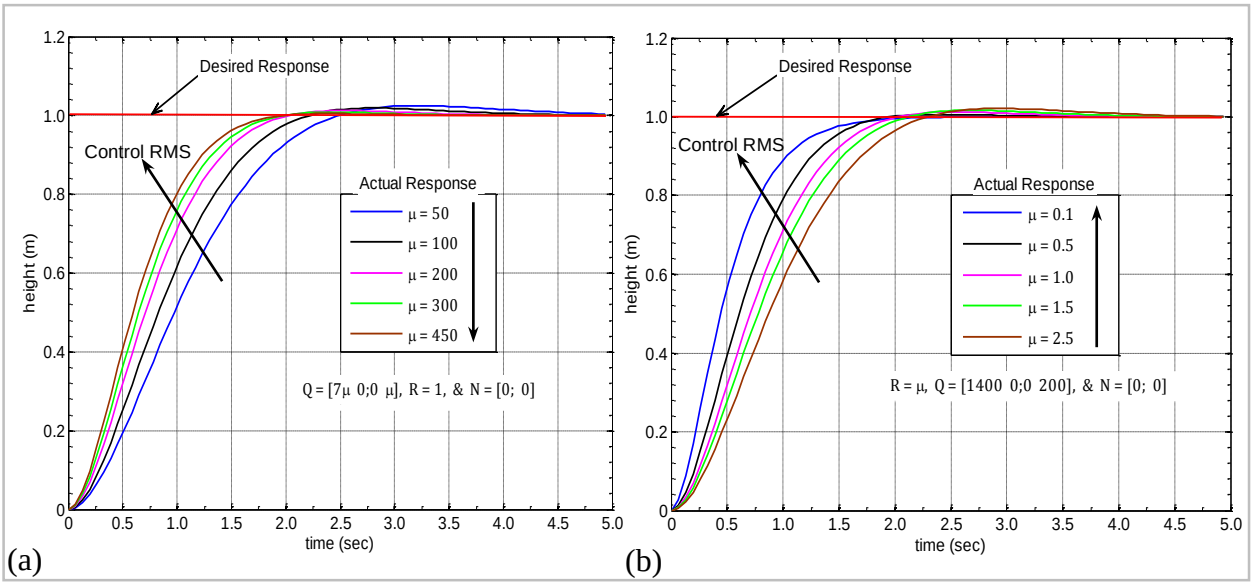


Figure 7.7. LQR responses without the nonlinear effects: (a) varying Q (b) varying R.

7.2.2 Pole placement with nonlinear effects: simulations. Figures 7.8a and 7.8b show the altitude and vertical speed responses by varying control input saturation. As the saturation values increase, the RMS value increases. Figure 7.9 also shows the altitude responses by varying T_d. The results show that as T_d increases, the RMS value increases. Also, at $T_d \cong 2.5T_s = 0.1625s$ the response starts to oscillate.

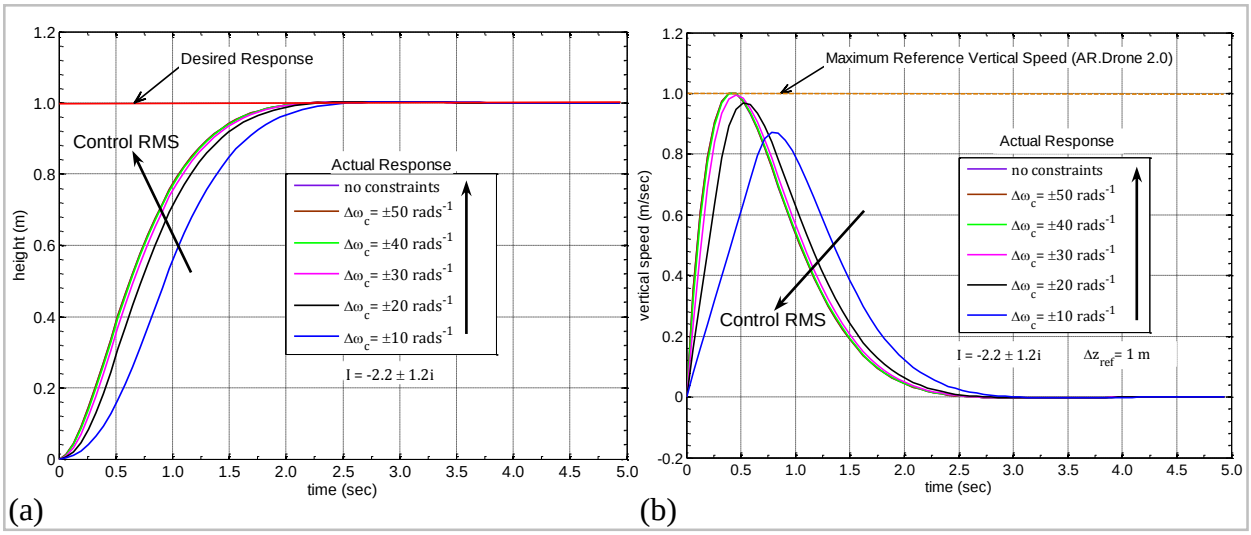


Figure 7.8. Pole placement (a) altitude (b) vertical speed: varying control input saturation.

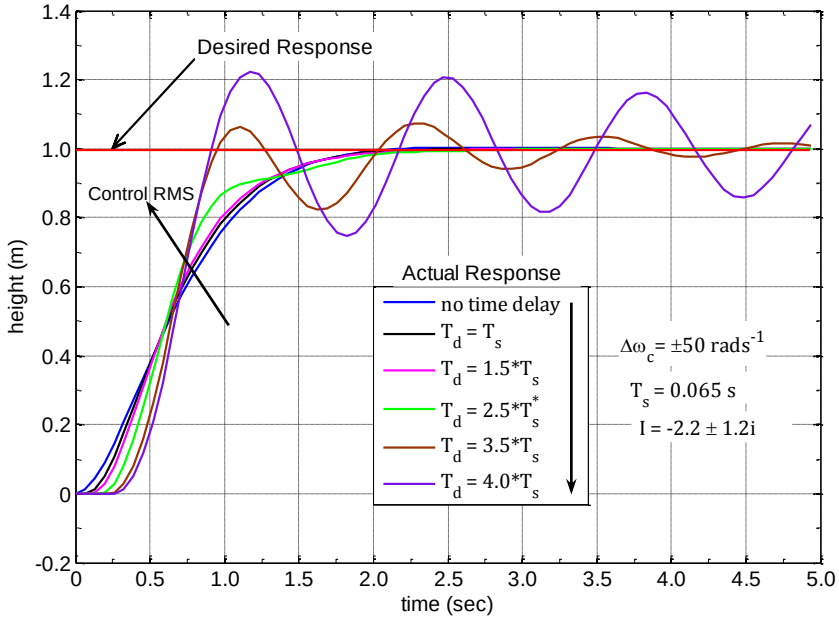


Figure 7.9. Pole placement altitude responses: varying time delay.

7.2.3 PD-MRAC with and without nonlinear effects: simulations. Figure 7.10 shows the altitude responses without time delay effects and PD controller by varying γ. It can be seen that the system did not stabilize. As mentioned in *Section 2.3.3.3.1*, the MIT rule MARC does not guarantee stability of the system, and therefore it was combined with PD controller.

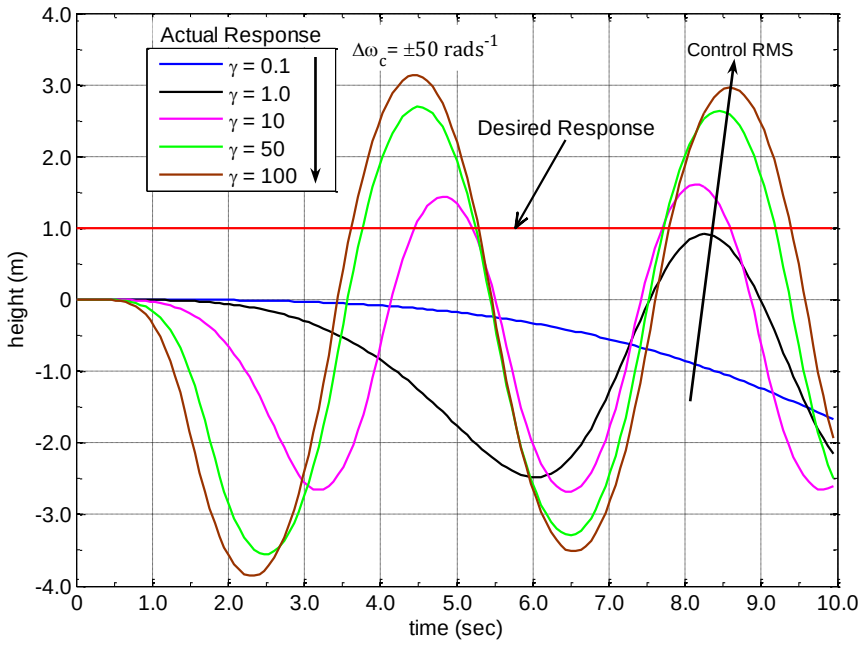


Figure 7.10. MRAC altitude response without time delay effects and PD controller.

Table 7.7 shows the transient response properties and the RMS values by varying γ. Figures 7.11a and 7.11b show the corresponding plots of the altitude responses and error signals, respectively. It can be seen that as γ increases, M_o, t_s, and t_p decreases but increases again, and the RMS value increases. Also, when $\gamma > 4210$ the response oscillates, and this is because the adaptation mechanism tries to quickly send the error signal to zero, which results in the oscillations. Moreover, when $\gamma < 1250$ the error signal takes longer time to go to zero, resulting slower response and higher $M_o\gamma$ has to be selected within a range of values, i.e., $1250 <\approx \gamma \approx< 4210$.

Table 7.7

PD-MRAC Altitude Responses: Varying γ

γ	M_o (%)	t_s (s)	t_p (s)	E_{ss} (m)	Control Energy (RMS Value)
100	4.66	3.47	2.54	-0.0022	7.24
1250*	1.99	1.70	2.28	8.73×10^{-4}	10.87
3000	1.10	1.76	2.34	-6.66×10^{-4}	11.67
4210*	1.17	1.78	2.28	9.85×10^{-4}	11.86
15000	3.31	3.89	2.60	0.0040	15.41

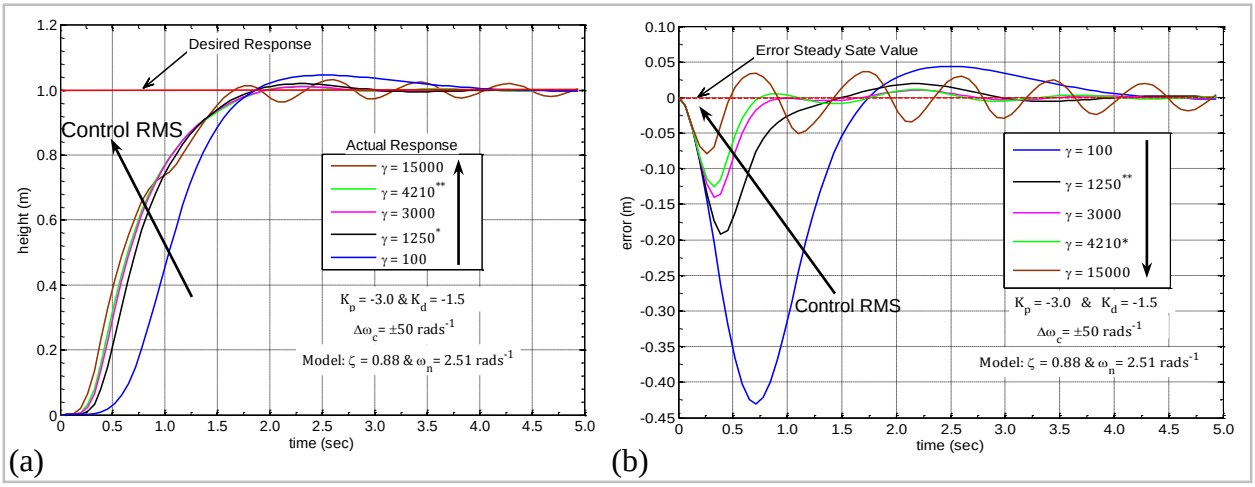


Figure 7.11. PD-MRAC (a) altitude (b) error: varying γ.

Figures 7.12a and 7.12b show the plots of the control input and vertical speed, respectively, by varying γ. As γ increases, the RMS value increases. The plots also show the saturation limits for the control input and the vertical speed. It is observed that $\gamma \cong 1250$ should be okay to satisfy the saturation limits, even though slightly higher values of γ can be used. Figures 7.13a and 7.13b show the plots of the altitude responses and vertical speed, respectively, of the PD-MRAC

controller as against using only the PD controller. The PD controller on its own did not stabilize the system, just like the MRAC on its own.

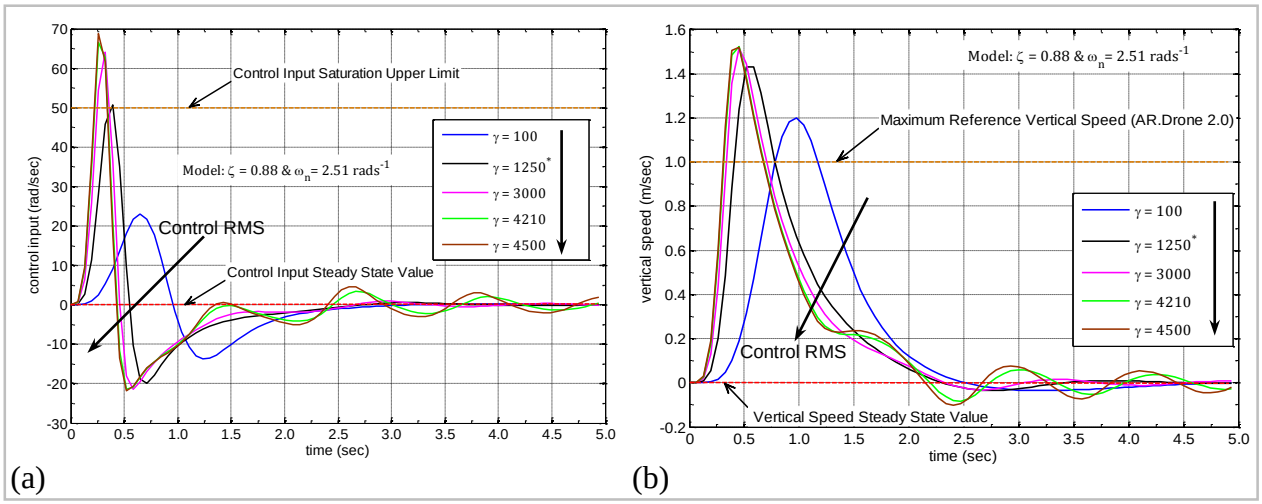


Figure 7.12. PD-MRAC (a) control input (b) vertical speed: varying γ.

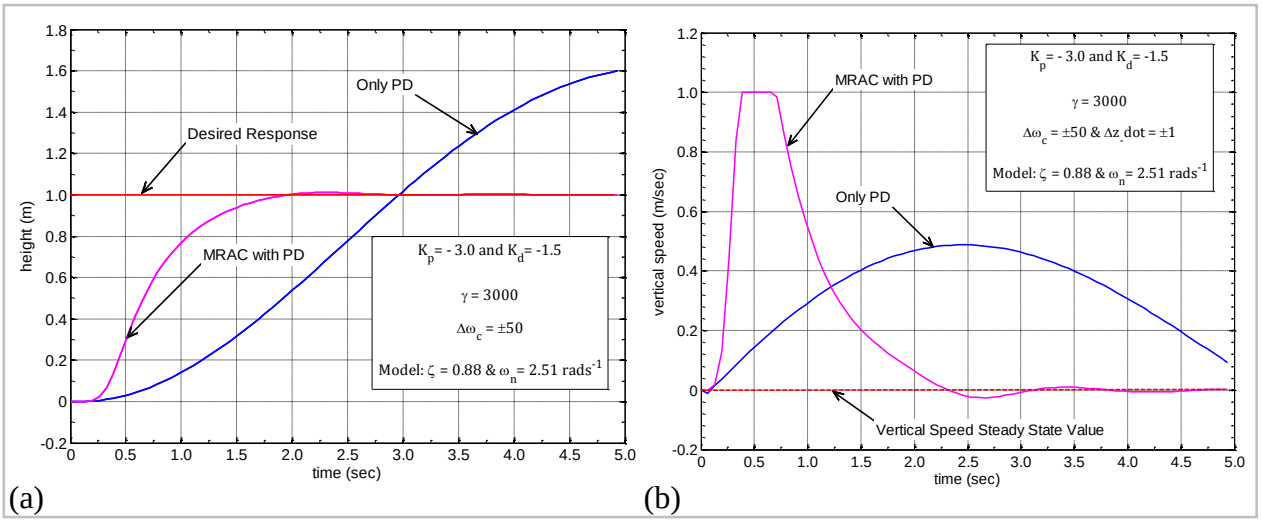


Figure 7.13. PD versus PD-MRAC (a) altitude (b) vertical speed.

Figure 7.14 shows the altitude responses by varying T_d. It can be seen that as T_d increases, the RMS value increases. Moreover, even at smaller values of T_d the response oscillates, indicating that the MRAC is very sensitive to the time delay effects.

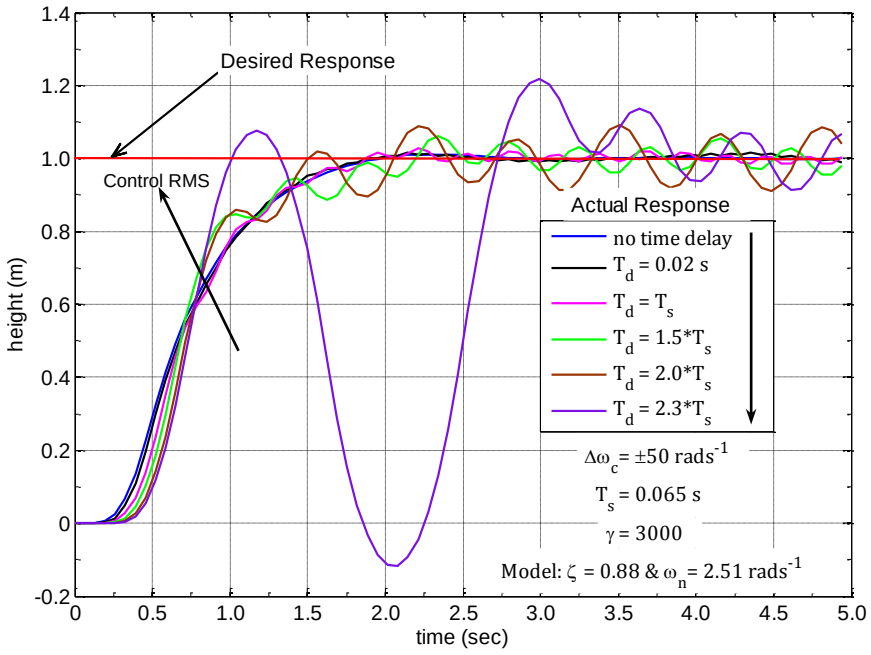


Figure 7.14. PD-MRAC altitude response with saturation: varying T_d.

7.2.4 Experiments. The approach of sending the reference vertical speed from the model to the drone encountered difficulties (see the setup in Figure 5.15). The drone did not stabilize to the desired height (e.g., see Figure 7.15). However, using an initial tuned P, PD, and PV controllers the system performance successfully improved in terms of time-domain specification, by sending the control signal directly to the drone (see the setup in Figure 5.14), instead of through the model, (e.g., see Figure 7.16).

The PD-MRAC never achieved stabilization, either by sending the control signal through the model or directly to the drone, but the pole placement controller, in the form of the PV controller, did achieve stabilization. MATLAB fixed-step higher-order solvers of *ode8 (Dormand-Prince)* and *ode14*x *(extrapolation)* were used, since it was observed that the type of solvers has significant effect on the type of controller, and thus the system response (see *Sections 7.4.1.1* for more on the effects of solvers). Note, for ease of analyzing the response, the experimental altitude responses, in some situations, has been shifted to start at $(0s, 0m)$. Now, in the meantime while

the stability issue with the MRAC implementation was figured out, the time delay in the drone's control system was estimated and its effects analyzed by designing PV and PV-MRAC controllers.

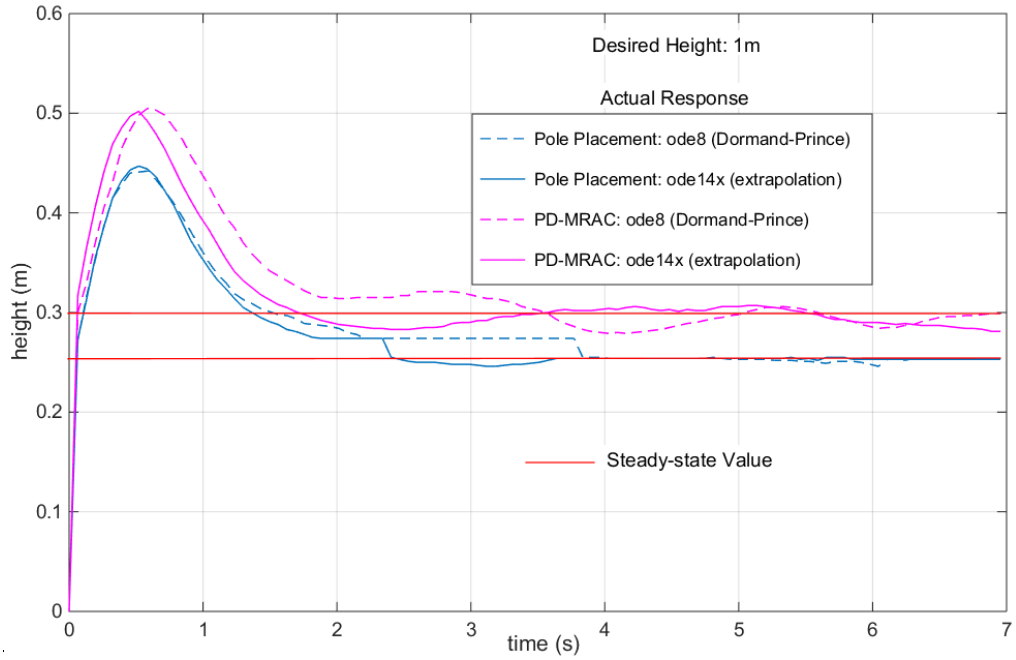


Figure 7.15. Pole placement and PD-MRAC responses: control signal through model.

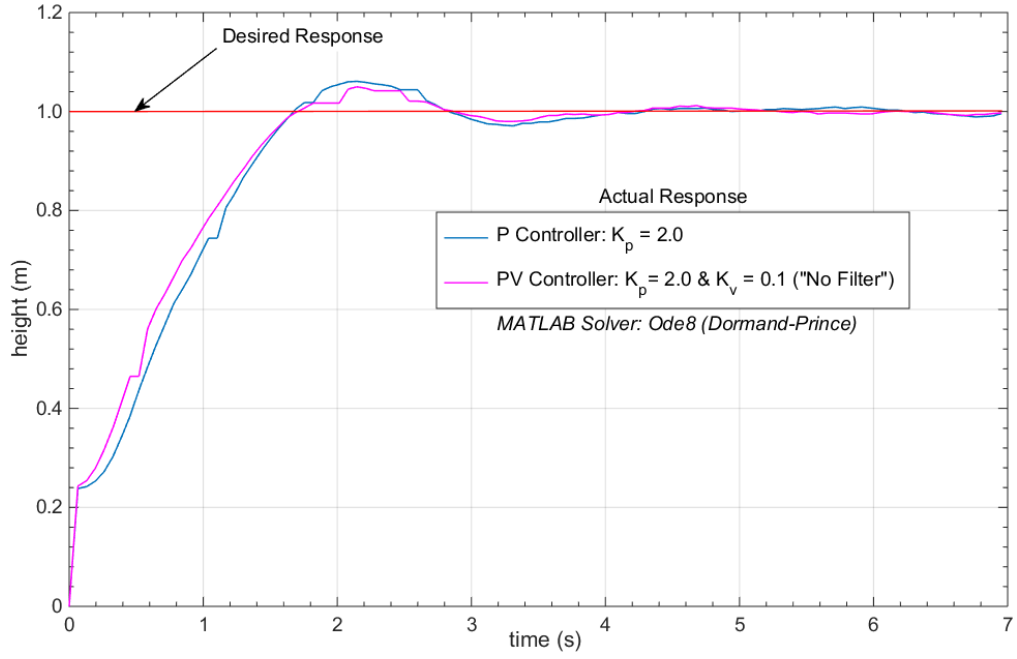


Figure 7.16. P and PV responses: direct control signal to drone.

7.3 Estimation of Time Delay: First-order Model

Initially, the drone's altitude responses were obtained for different values of K_p, as shown in Figure 7.17. Note that if there is no delay ($T_d = 0$), there should be no overshoot. The characteristic root is $-K_p$, refer to (3.106), which is a real number. However, as seen in Figure 7.17, the delay introduces imaginary parts in the roots and, thus, oscillation in the responses. Therefore, the delay has to be estimated and considered in designing control systems.

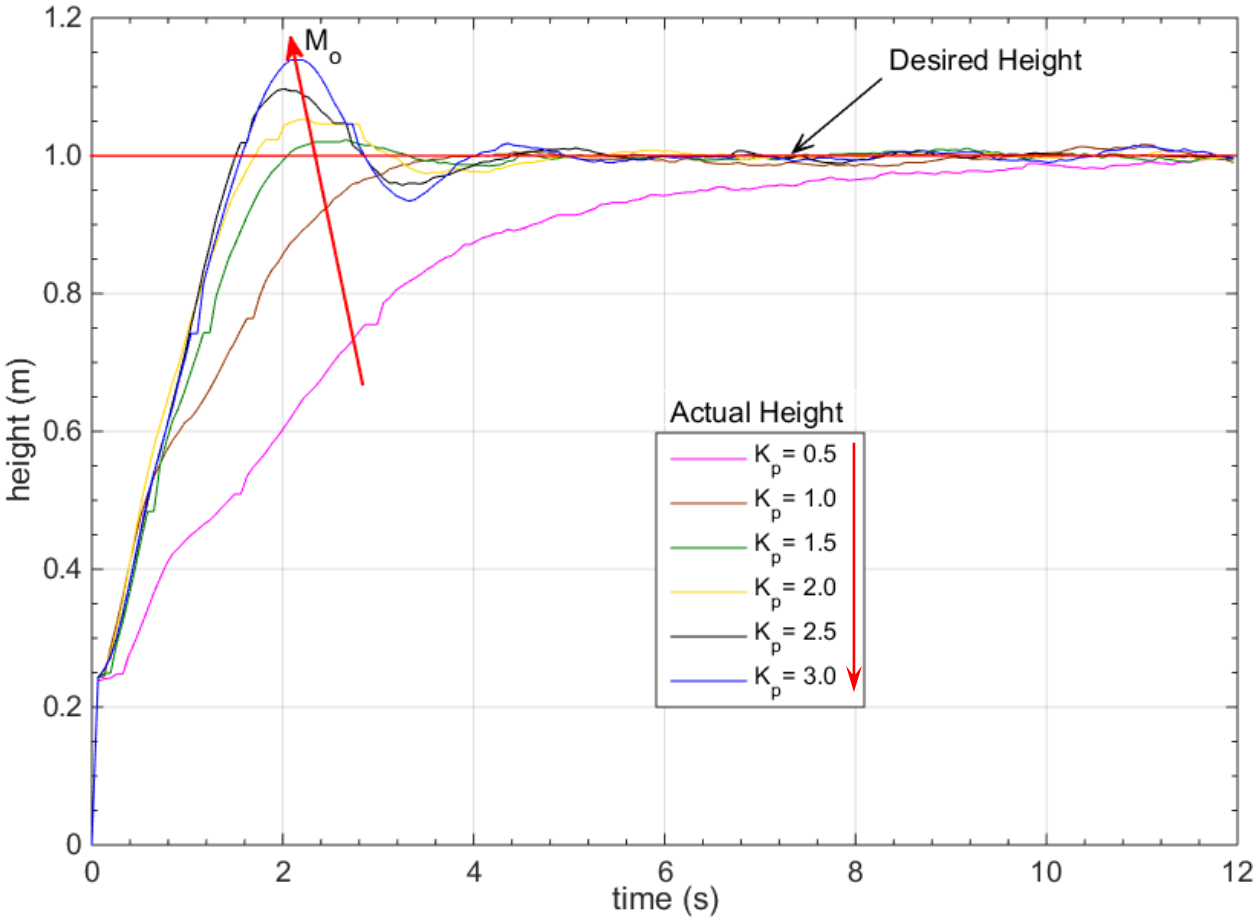


Figure 7.17. Experimented P controller altitude responses: varying K_p.

The responses in Figure 7.17 shows that $K_p = 1.0$ seems to be ideal for the controller since the response has no overshoot; however, the response is very slow with a settling time of about $10s$. As it can also be observed, increasing K_p makes the response faster, the rise time becomes

shorter, but introduces higher M_o. This is partly due to the time delay in the system, which introduces nonlinearity on the dynamics. It was also observed that the saturation applied to the control input has a nonlinear effect on the system's response, especially as K_p increases.

Table 7.8

Simulated Effect of Control Input Saturation on the Altitude Response

	Without Saturation		With Saturation	
	M_o (%)	t_p (s)	M_o (%)	t_p (s)
$K_p = 1.00$ & $T_d = 5T_s$	0.00	6.96	0.00	6.96
$K_p = 3.00$ & $T_d = 4T_s$	27.78	0.91	9.42	1.56
$K_p = 1.31$ & $T_d = 5T_s$	0.44	2.29	0.42	2.33

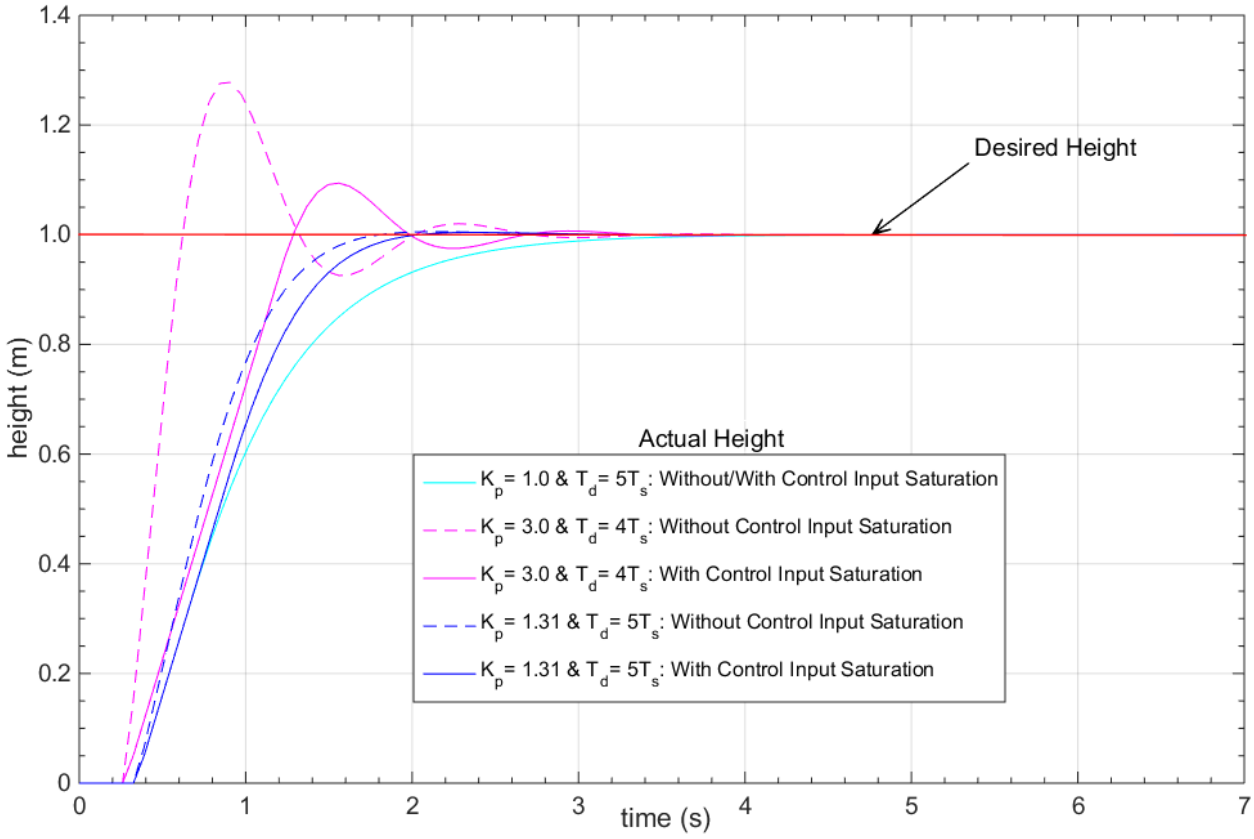


Figure 7.18. Simulated effect of control input saturation on the altitude response.

Investigation through simulations revealed that at a constant T_d, as the K_p value increases the control input saturation has a significant effect on the system response, see Figure 7.18 and Table 7.8. It can be observed that, at a constant T_d and K_p values, introducing the saturation decreases M_o and increases t_p. As K_p increases the increase in M_o becomes large and the oscillations of the system increase. Also, it can be seen that for $K_p = 1.0$ at $T_d = 5T_s$, there is no overshoot, as compared to $K_p = 3.0$ at a lower $T_d = 4T_s$, with and without the saturation effects. Moreover, it can be observed from the experimented altitude responses that at $K_p = 1.0$, there is no or little overshoot, and the oscillations are very small (see Figure 7.17); thus, the system's time delay did have little effect on the transient response. Using simulation, an appropriate $K_p = 1.31$ was selected with $T_d = 5T_s$, that gives a response with a sufficient overshoot for estimation, and with minimum saturation effect (see Figure 7.18 and Table 7.8).

7.3.1 Experimental method. Table 7.9 shows a summary of the simulation altitude responses transient properties, by varying T_d at $K_p = 1.31$, where K is a real constant tuning parameter. Samples of the drone altitude responses with $K_p = 1.31$ are shown in Figure 7.19, with Table 7.10 displaying their corresponding M_o and t_p values. The value, $t_p = 3.055s$, with the highest $M_o = 2.300\%$ gives the largest T_d. Comparing the $M_o = 2.300\%$ to the results in Table 7.9, T_d is estimated as $5.6646T_s$, which gives $0.368s$.

7.3.2 Analytical method: use of characteristic roots. The drone altitude response oscillates (Figures 7.17 and 7.19). Thus, the system has two complex conjugate dominant roots, and therefore, M_o and t_p was used to determine ξ and ω_n. $M_o = 2.300\%$ and $t_p = 3.055s$ (see *Section 7.3.1*), thus, ξ and ω_n are computed as 0.7684 and $1.6069rads^{-1}$, respectively using (2.15). Using (3.110), the dominant characteristic roots, approximated, are calculated as $s =$

$-1.2347 \pm 1.0284j$. Then, from (3.109), T_d is determined as $0.374s$ using *fsolve* in MATLAB with initial guess value of 0.2s. See Figure 7.20 for the iteration of the *fsolve*.

Table 7.9

Simulated Altitude Responses: K_p = 1.31

K	$T_d = KT_s$ (s)	M_o (%)	t_p (s)
4.0000	0.260	0.000	6.955
5.0000	0.325	0.419	2.340
5.6000	0.364	2.067	2.080
5.6640	0.368	2.298	2.080
5.6645	0.368	2.301	2.080
5.6646*	0.368*	2.300*	2.080*
5.6660	0.369	2.305	2.080
5.7000	0.371	2.429	2.080

Table 7.10

Experimented Altitude Responses: K_p = 1.31

	Flight				
	1	2	3	4	5
M_o (%)	2.300*	2.290	2.300	2.270	2.140
t_p (s)	3.055*	3.084	3.575	3.194	3.096

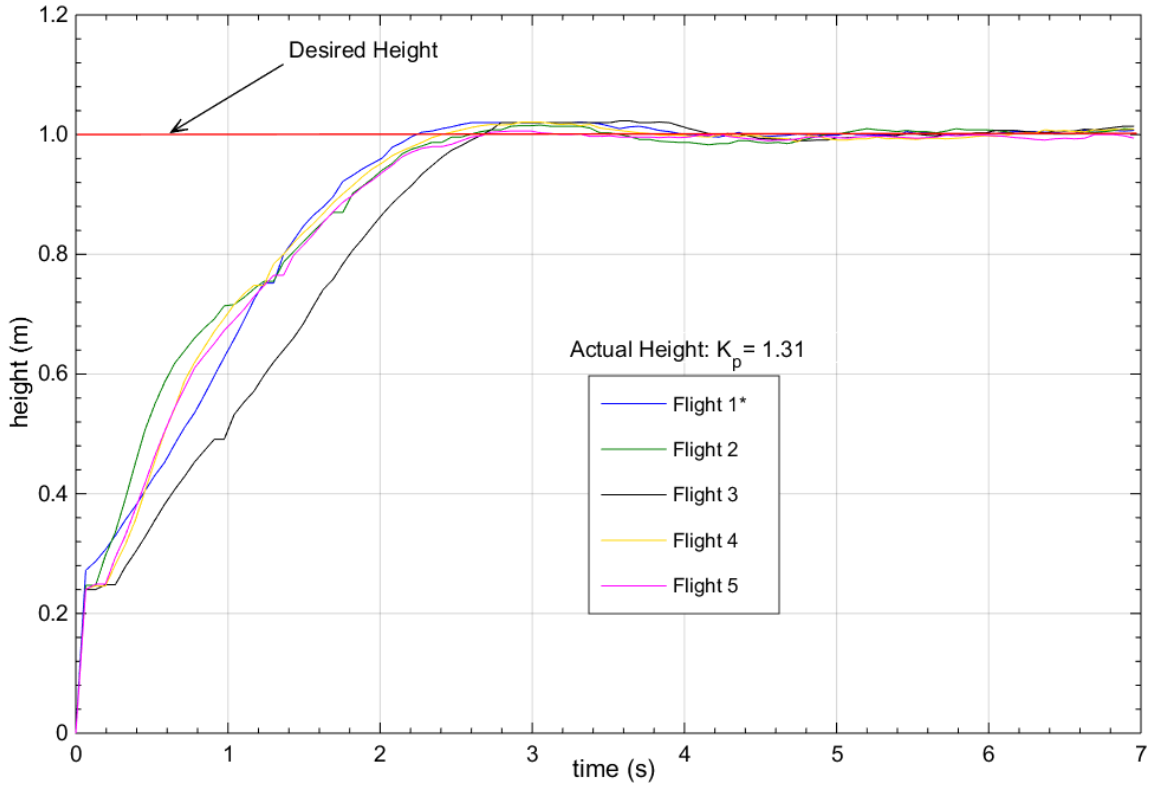


Figure 7.19. Experimented P controller altitude responses: $K_p = 1.31$.

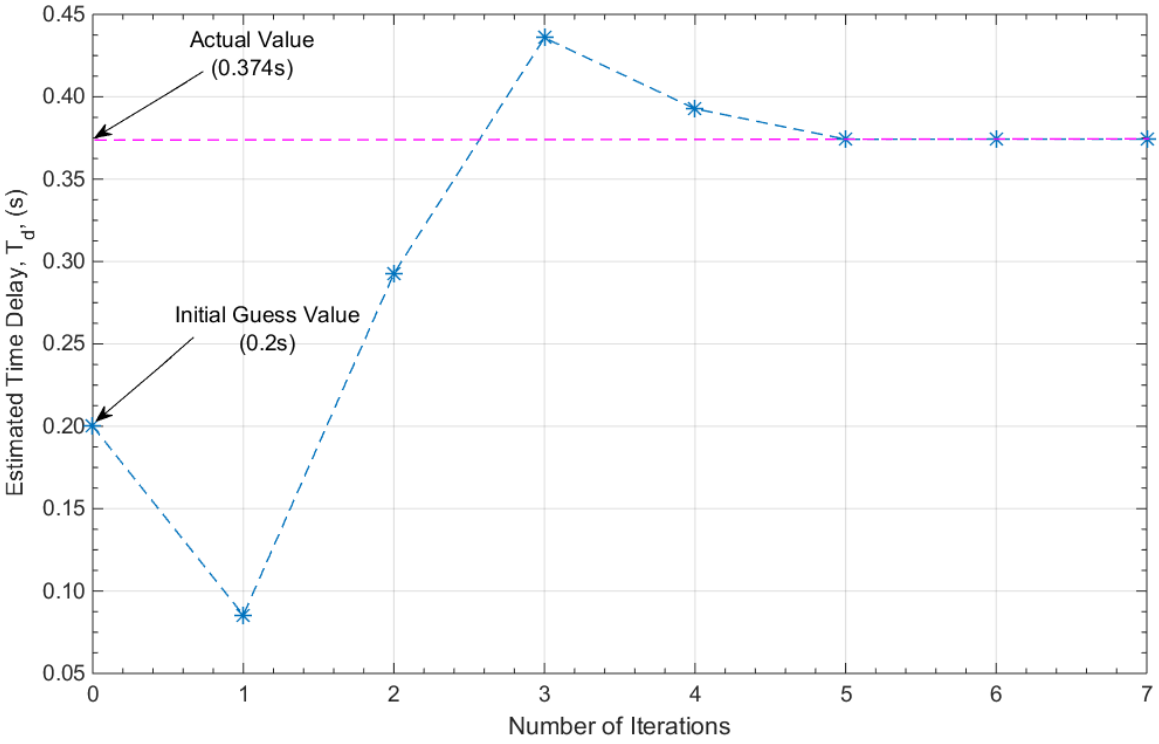


Figure 7.20. Iteration of MATLAB *fsolve* to estimate the time delay.

7.4 Designed Controllers: First-order Model

The drone control system estimated time delay, using both the experimental and the analytical methods, obtained was approximately 0.37s (*see Sections 7.3.1* and *7.3.2*).

7.4.1 PV controller. A MATLAB-based software package [78] was used to study the stability of the neutral-type time-delay system, by solving the characteristic equation from the transfer function in (3.111). The closed-loop system characteristic roots within a specified region are then plotted for various K_v values.

Figure. 7.21 shows the spectrum distribution of the characteristic roots, and Table 7.11 shows a summary of the rightmost (i.e., dominant) roots for each system. The value $K_v = 0.3$ yields the most suitable and stable rightmost roots among them. The corresponding simulation altitude responses for the system were also obtained for the various K_v values, see Figure. 7.22. It can be seen that as K_v increases, at $K_p = 2.0$ and $T_d = 0.37\text{s}$, M_o decreases and the rise time becomes longer. At higher values of K_v, the response oscillates and the system becomes unstable. This is also observed in Figure 7.21, that as K_v increases the roots move to the right, increasing the instability in the system.

Table 7.11

Rightmost Characteristic Roots of the PV Control System: First-order Model

K_v	Rightmost Complex Roots
0.0	-1.42 ± 3.07j
0.1	-1.98 ± 3.25j
0.3*	-3.25 ± 24.75j
0.5	-1.84 ± 6.98j
0.7	-0.89 ± 7.47j

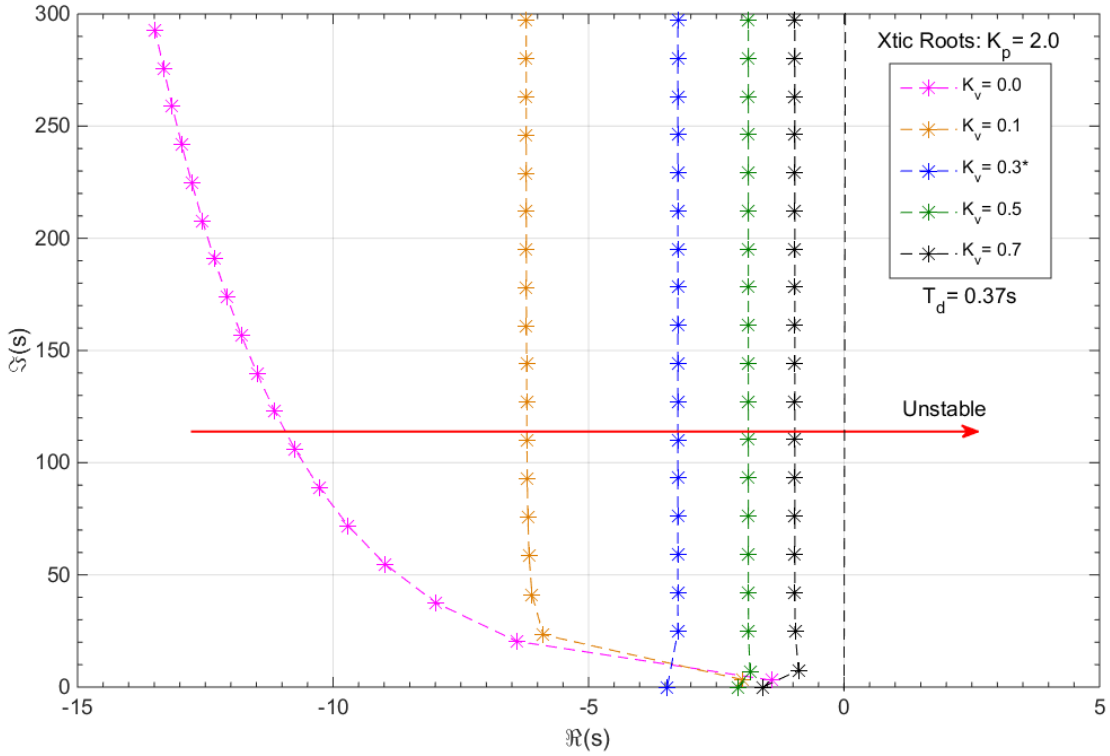


Figure 7.21. PV control system characteristic roots spectrum distribution: first-order model.

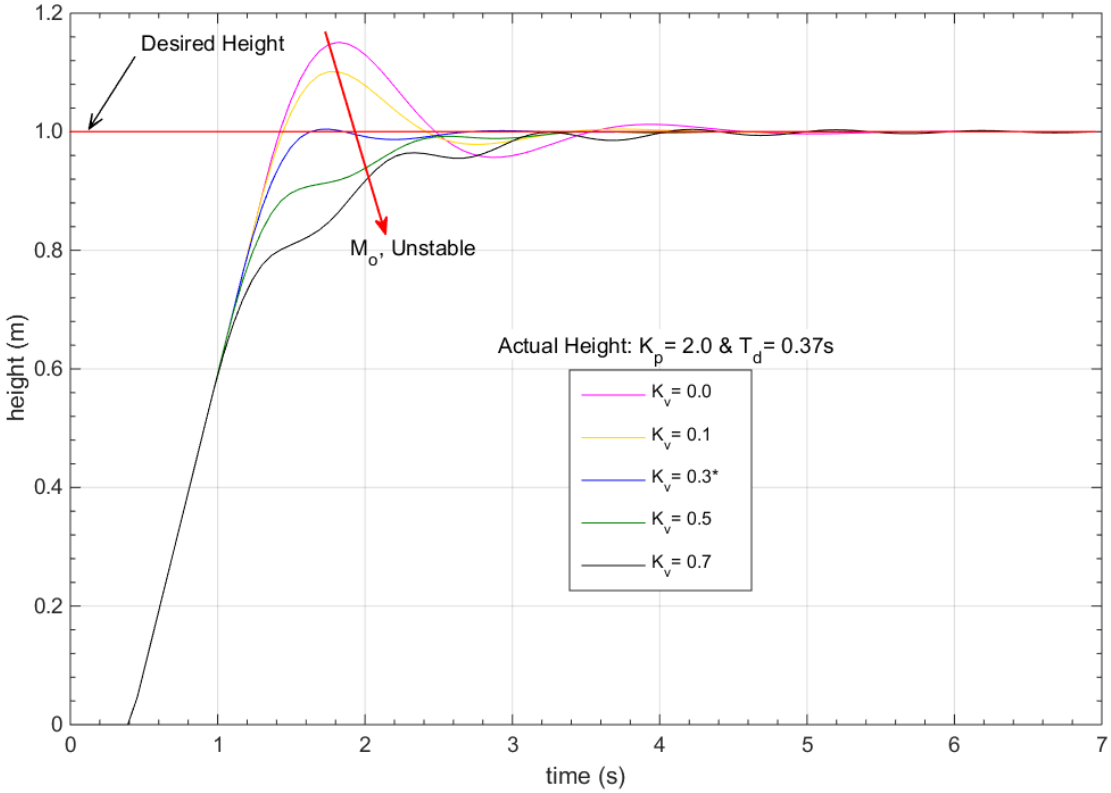


Figure 7.22. Simulated PV controller altitude response without HPF: first-order model.

Table 7.12

Simulated Effect of the HPF on the Response, Varying ω_f: First-order Model

	ω_f (rads^{-1})				
	5	20	38*	50	70
M_o (%)	3.19	0.64	0.32	0.38	15.09
t_p (s)	3.32	2.93	1.69	1.76	2.54
t_s (s)	4.24	2.43	1.52	1.52	NaN

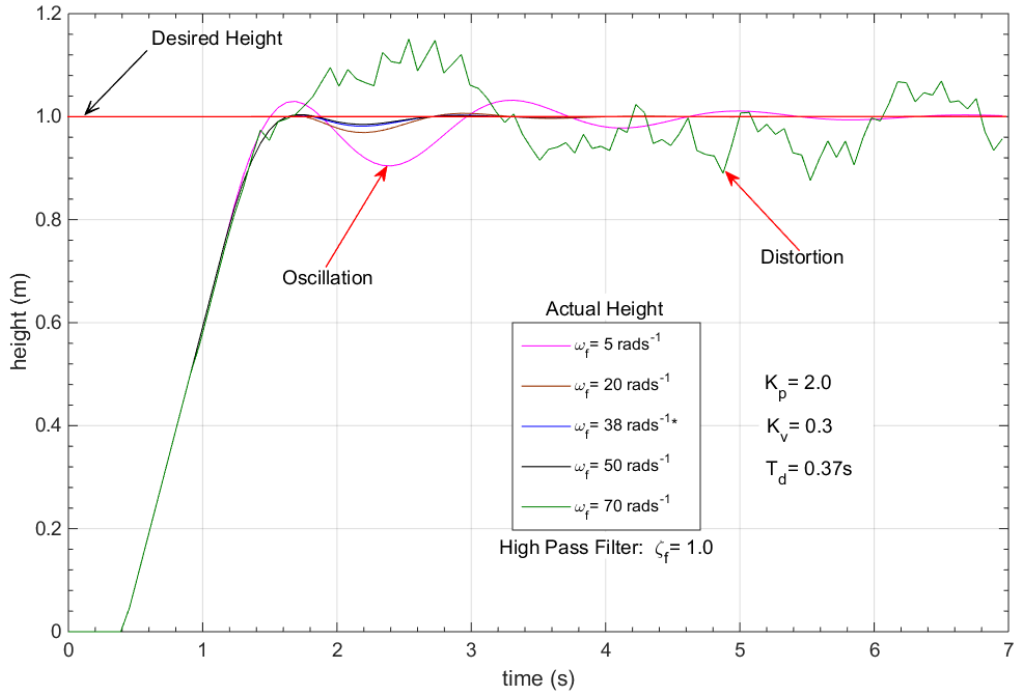


Figure 7.23. Simulated effect of the HPF on the response, varying ω_f: first-order model.

Now, based on these analysis, a controller with $K_p = 2.0$ and $K_v = 0.3$ was selected as the most suitable, with closed-loop system response transient properties of $M_o = 0.44\%$, $t_s = 1.52$s, and $t_p = 1.76$s. Using these controller gains, an HPF was included in the simulation control system, and its effects on the altitude transient response, at different ω_f, was studied, see Figure 7.23 and Table 7.12. It is observed that at smaller ω_f values the response oscillates, and at higher

values the response distorts. The oscillations and the distortions effects were reduced by using the high-order solver, *ode8 (Dormand-Prince)*.

An HPF with $\omega_f = 38\ \text{rads}^{-1}$ and $\xi_f = 1.0$ was then selected, with poles of -38 repeated. Now, looking at the poles distribution of the system in Figure 7.21, it can be observed that the poles of this filter are located to the left than the poles of the PV-feedback closed-loop system, without the filter effect. Thus, this filter will respond faster, therefore, it will have smaller effect on the drone's altitude transient response. Bode plot (see Figure 7.24) was used to determine the filter's cutoff frequency as $5.68\ \text{rads}^{-1}$ (0.90 Hz).

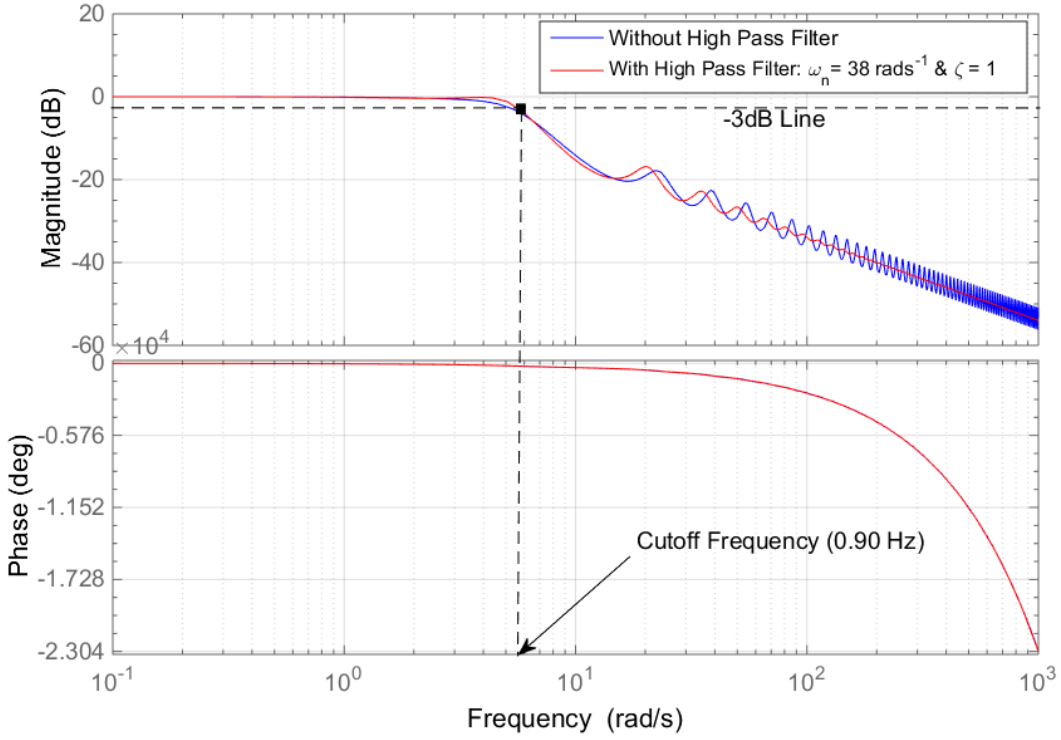


Figure 7.24. Bode plots of the PV-feedback close-loop system: first-order model.

Figure 7.25 and Table 7.13 shows the simulation altitude responses and their corresponding transient properties, with the HPF, and $K_p = 2.0$ for different K_v values. The results with $K_p = 2.0$ and $K_v = 0.3$ show an improved transient response performance, which suggests that the estimation of delay and analysis presented help. Figure 7.26 and Table 7.14 also shows the

experimented altitude responses and their corresponding transient properties, with the HPF, and $K_p = 2.0$ for different K_v values. It can be seen that as the K_v value increases M_o decreases, and in general the responses become slower. The PV controller performed better for $K_v = 0.3, 0.5,$ and 0.7 at $K_p = 2.0$.

Table 7.13

Simulated PV Controller Transient Properties with the HPF: First-order Model

	K_v				
	0.0	0.1	0.3*	0.5	0.7
M_o (%)	15.10	10.10	0.32	0.15	0.70
t_p (s)	1.82	1.76	1.69	3.64	4.36
t_s (s)	3.28	2.87	1.52	2.33	3.04

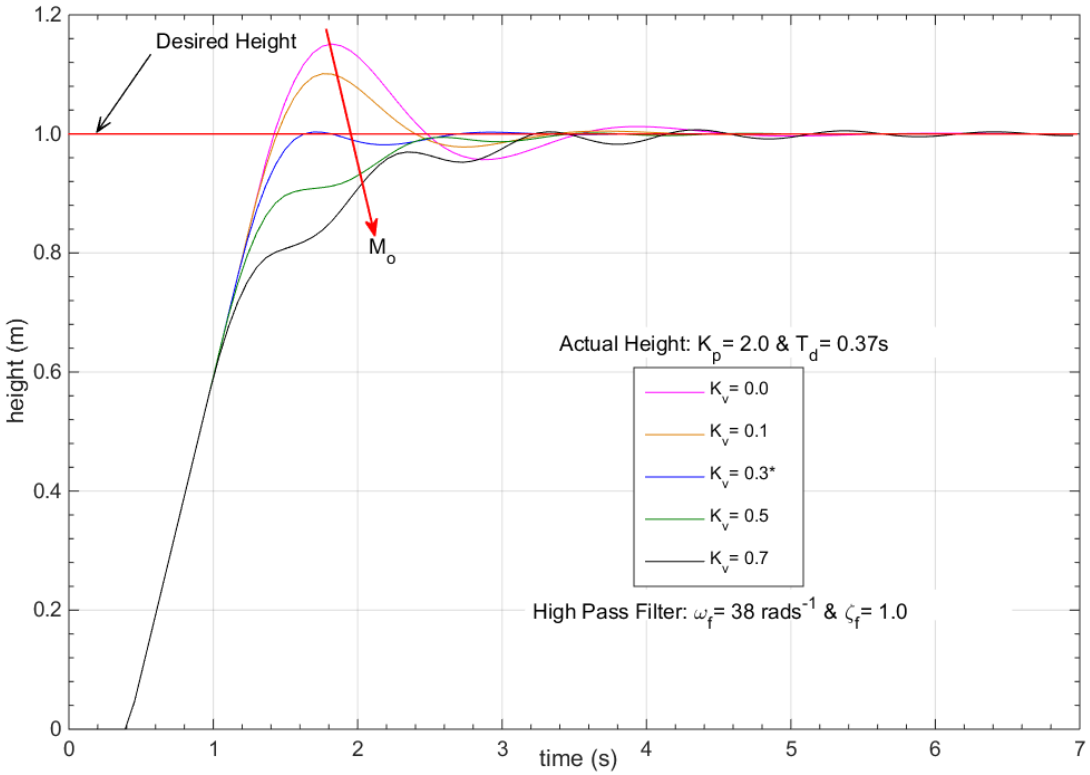


Figure 7.25. Simulated PV controller altitude responses with the HPF: first-order model

Table 7.14

Experimented PV Controller Transient Properties with the HPF

	K_v				
	0.0	0.1	0.3*	0.5	0.7
M_o (%)	8.40	5.30	2.92	0.80	0.40
t_p (s)	2.21	2.15	2.18	4.76	4.07
t_s (s)	4.00	2.67	2.86	1.99	2.48

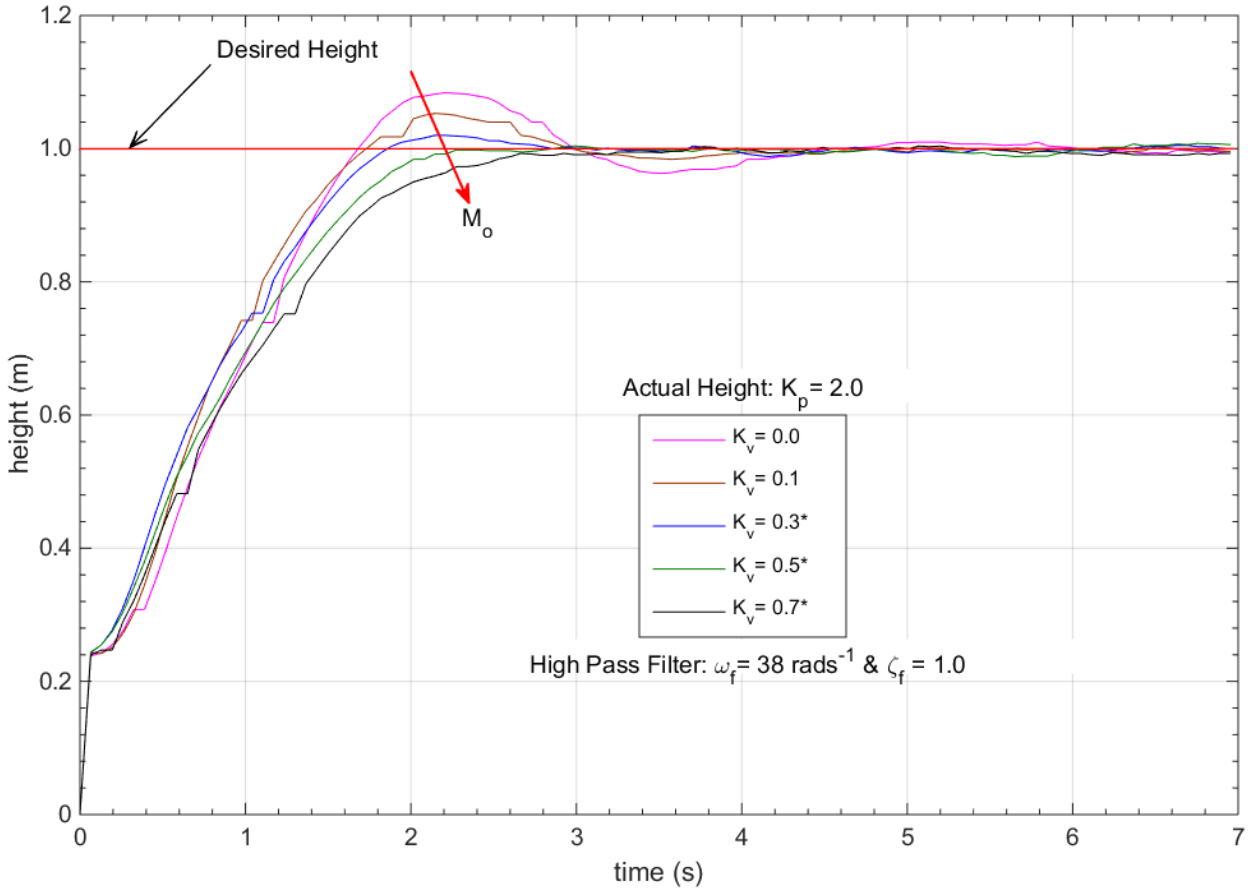


Figure 7.26. Experimented PV controller altitude responses with the HPF: first-order model

7.4.1.1 Effects of solvers on the response. The simulations and experiments were conducted using MATLAB/Simulink fixed-step solver, *ode8 (Dormand-Prince)*. The solver computes the model's states during simulation and code generation. Investigation revealed that the

choice of a solver has an effect on the drone altitude response, refer to Figure 7.27, and also see [79, 80] for more details. The HPF performance is constrained by the type of solver used; the filter performs better at higher ω_f with the high-order solvers.

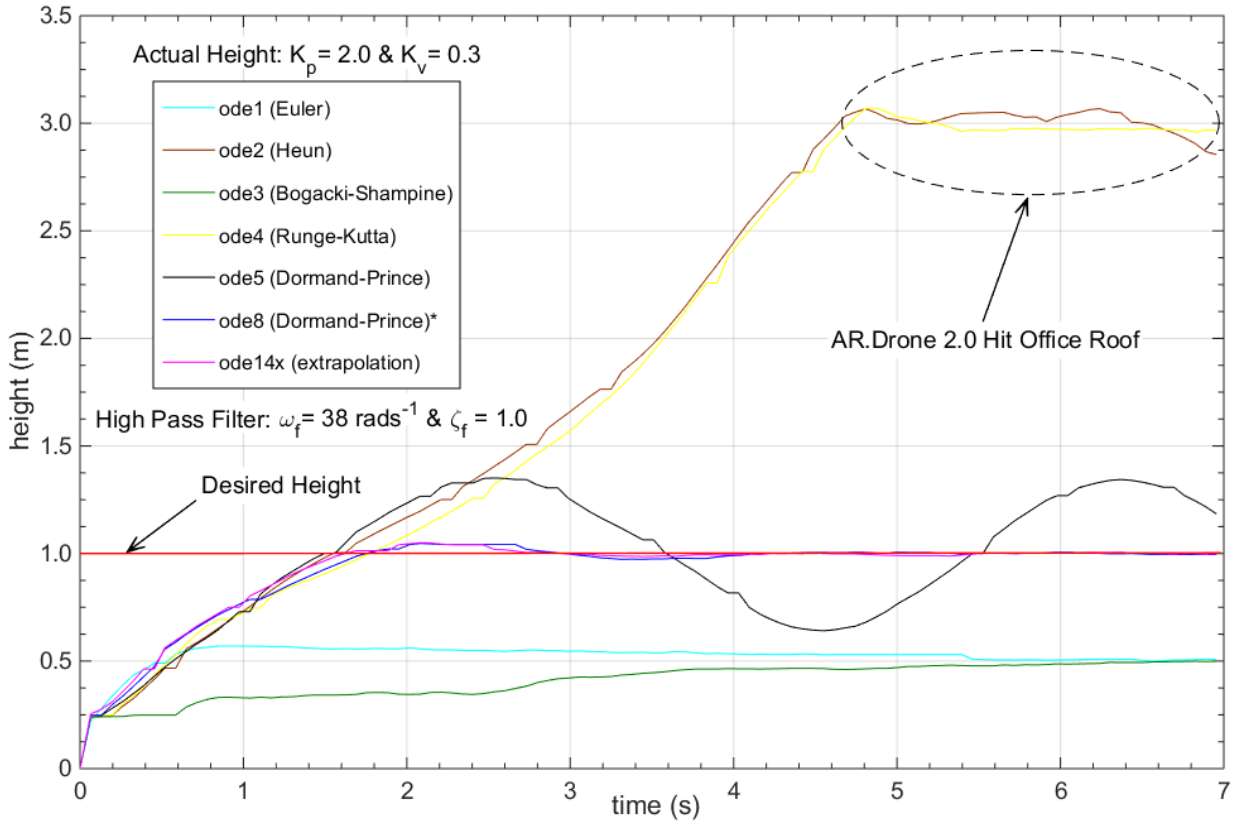


Figure 7.27. Experimented effect of Simulink solvers on the PV controller altitude response.

7.4.2 PV-MRAC controller. Figure 7.28 shows the simulation and experiment altitude responses of the MIT rule PV-MRAC controller considering the effects of control input saturation, and with and without the time delay in the control system. ode14x (extrapolation) solver was used. It can be seen that the response does not oscillates without the application of the time delay. As earlier investigation revealed (see Figure 7.14), the MRAC control system is very sensitive to time delay effects. Therefore, it can be seen that when the controller was applied in the drone's control system the response oscillates and never stabilized, which was the case for the simulation as well. This also confirms the fact that drone's control system contains time delay.

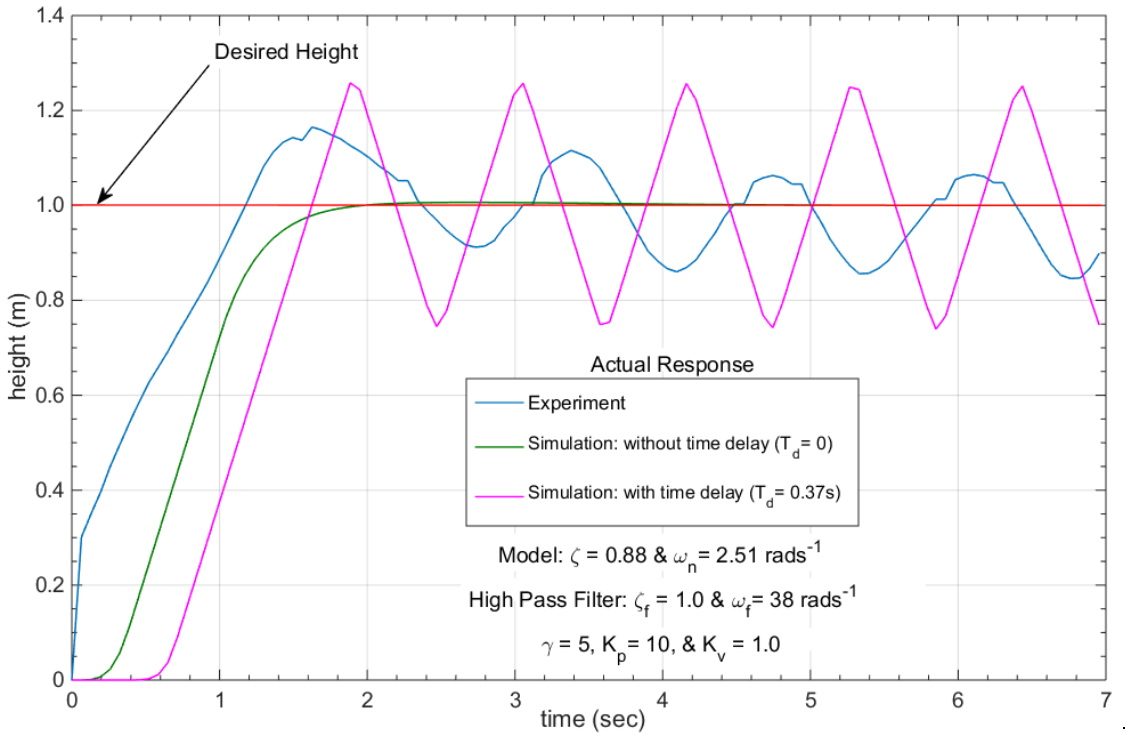


Figure 7.28. PV-MRAC controller altitude responses: first-order model

The stability issue could be addressed by: (1) Use of the right tuning parameters such as γ, K_p, K_v, ξ, and ω_n; this was done, but the response never stabilized. (2) Use a higher-order solver, and possibly a variable-step time; MATLAB has higher-order solvers for variable-step time such as *ode45 (Dormand-Prince)*, however, *Real-time windows target (RTW)* application requires fixed-step time, with the highest solver being *ode14*x *(extrapolation)*. (3) Use a higher specification PC; this was also considered (4) Develop a better altitude model; this was done using the black-box approach, MATLAB system identification toolbox, see *Section 7.5* for the results.

7.5 Designed Controllers: Black-box Approach (Second-order Model)

Figures 7.29 to 7.31 shows the altitude responses from experiment and simulation with and without the time delay effects for the P, PV, and PV-MRAC controllers, respectively. The *ode8 (Dormand-Prince)* solver was used for the P and PV control systems. Initial results show that the *ode8* solver could not handle the PV-MRAC simulation control system even without time delay

effects. Also, the *ode14*x *(extrapolation)* did handle the PV-MRAC simulation and experiment control systems to an extent. For the simulation, the solver performs well when $T_d \in [0 \;\; 0.04993]s$ with some transient response oscillations, and for the experiment the response oscillated about the desired height and never stabilized (see Figure 7.31).

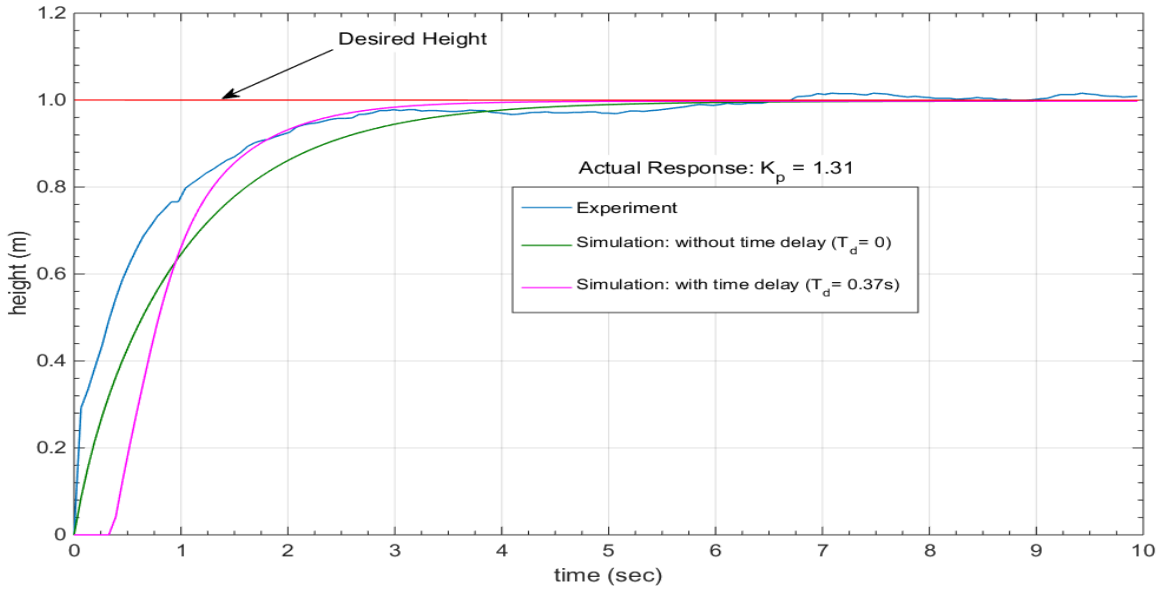


Figure 7.29. P controller altitude responses: black-box approach model.

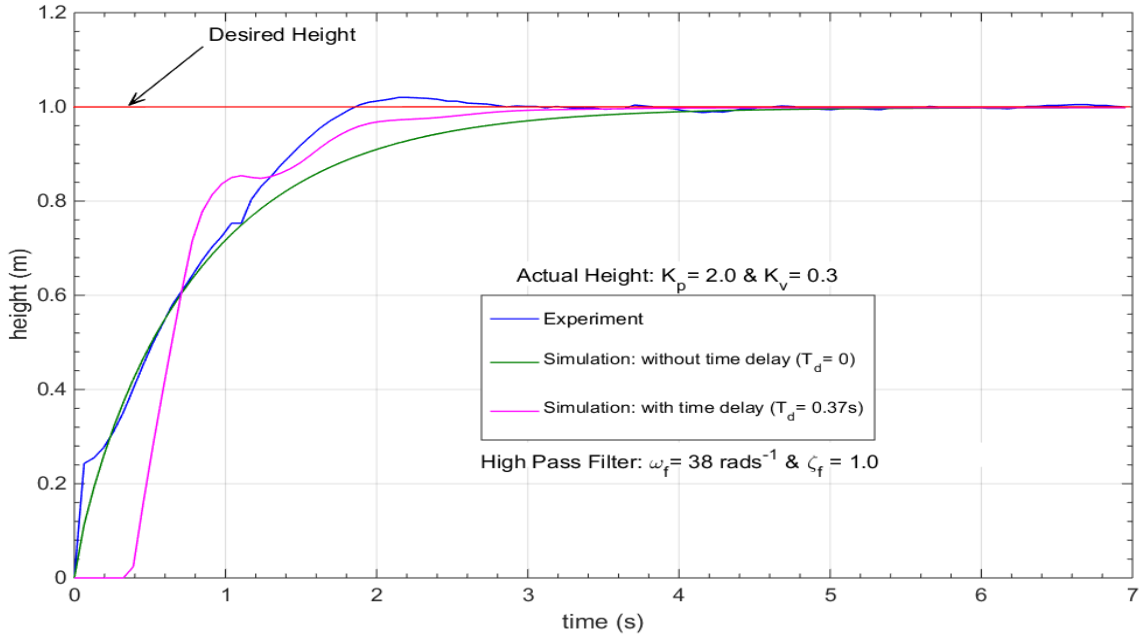


Figure 7.30. PV controller altitude responses: black-box approach model.

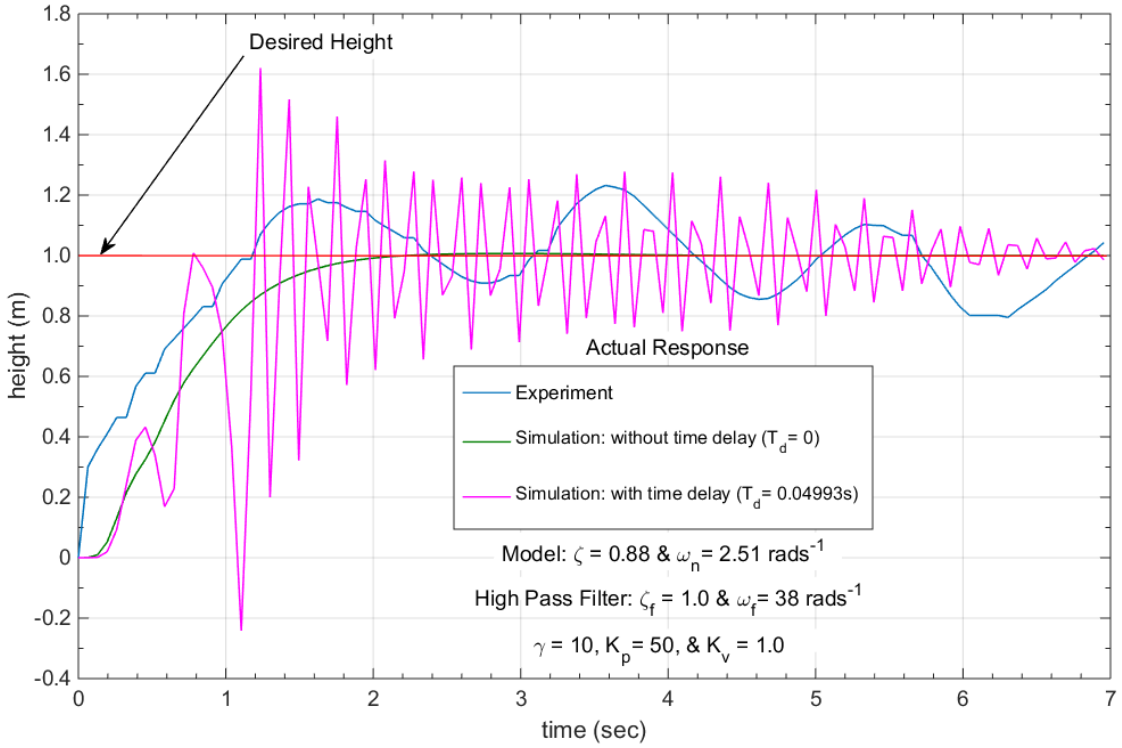


Figure 7.31. PV-MRAC controller altitude responses: black-box approach model.

The simulation control system finds it difficult to stabilize for $T_d \geq 0.04993s$. Therefore, when the controller was applied in the drone's control system the response oscillates and never stabilized, which confirms the fact that the time delay in the control system is definitely bigger than $0.04993s$. The stability issue could now be addressed by the following options: (1) improving on the tuning parameters of γ, K_p, K_v, ξ, and ω_n and/or (2) using a higher specification PC.

Stabilization was finally achieved by obtaining the right tuning parameters. Figures 7.32 and 7.33 show simulated and experimented responses, respectively, by using different γ values. It can be seen that increasing γ makes the response faster but it introduces oscillations, while decreasing γ makes the response slower leading to an unstable system. The results confirms earlier conclusion (see Table 7.7) that there is a range of values of γ to achieve stabilization, assuming the other tuning parameters remains constant. The results in Figure 7.34 and Table 7.15 compares simulated and experimented with and without the time-delay effects.

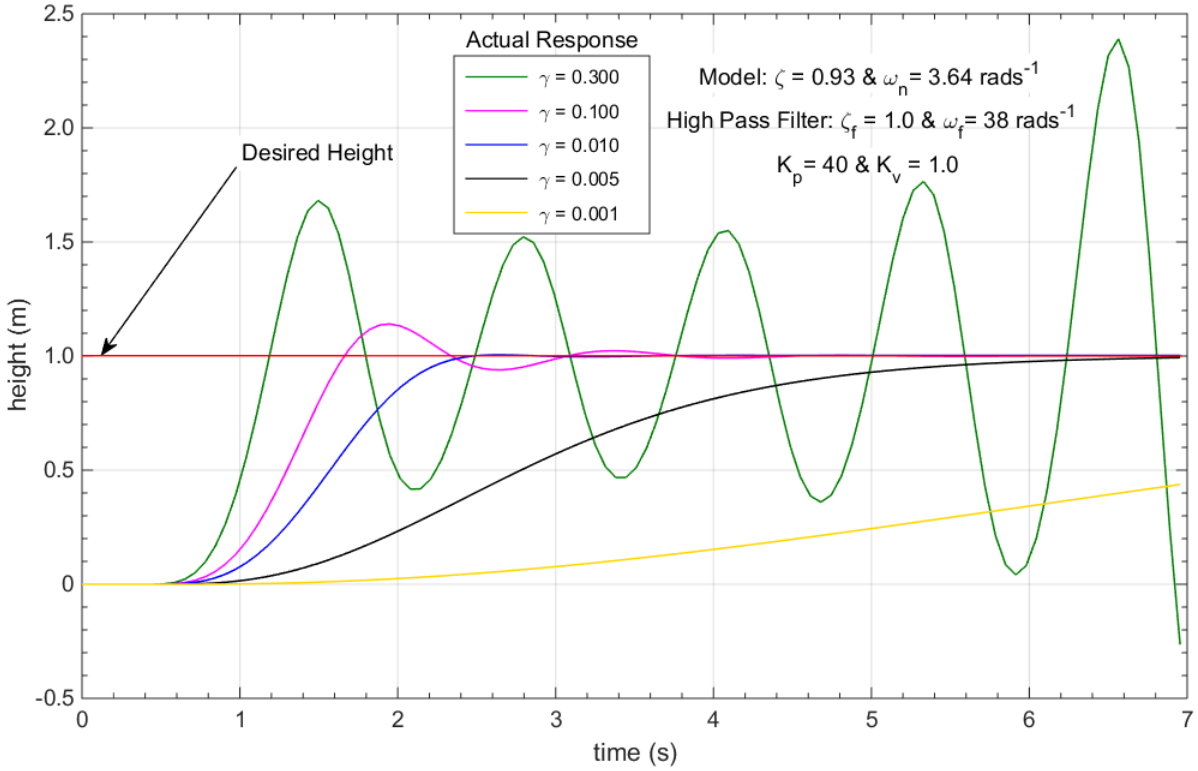


Figure 7.32. Simulated PV-MRAC responses, varying γ: black-box approach model.

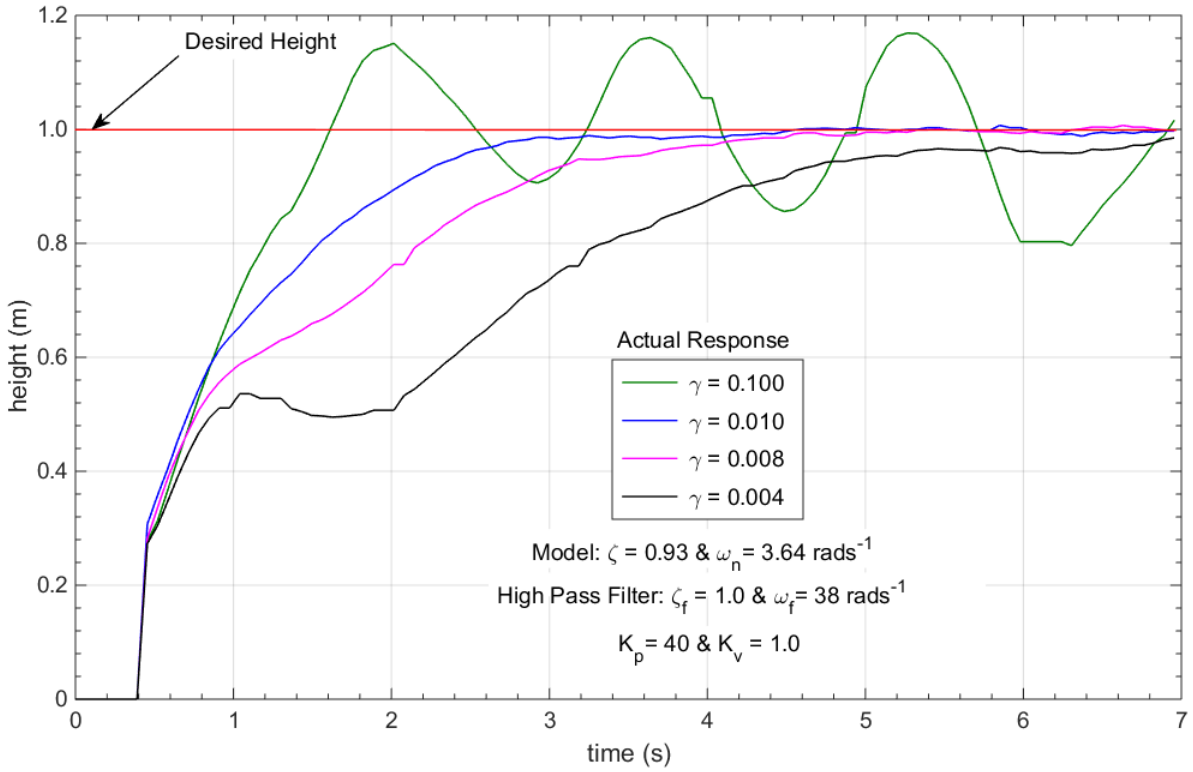


Figure 7.33. Experimented PV-MRAC responses, varying γ: black-box approach model.

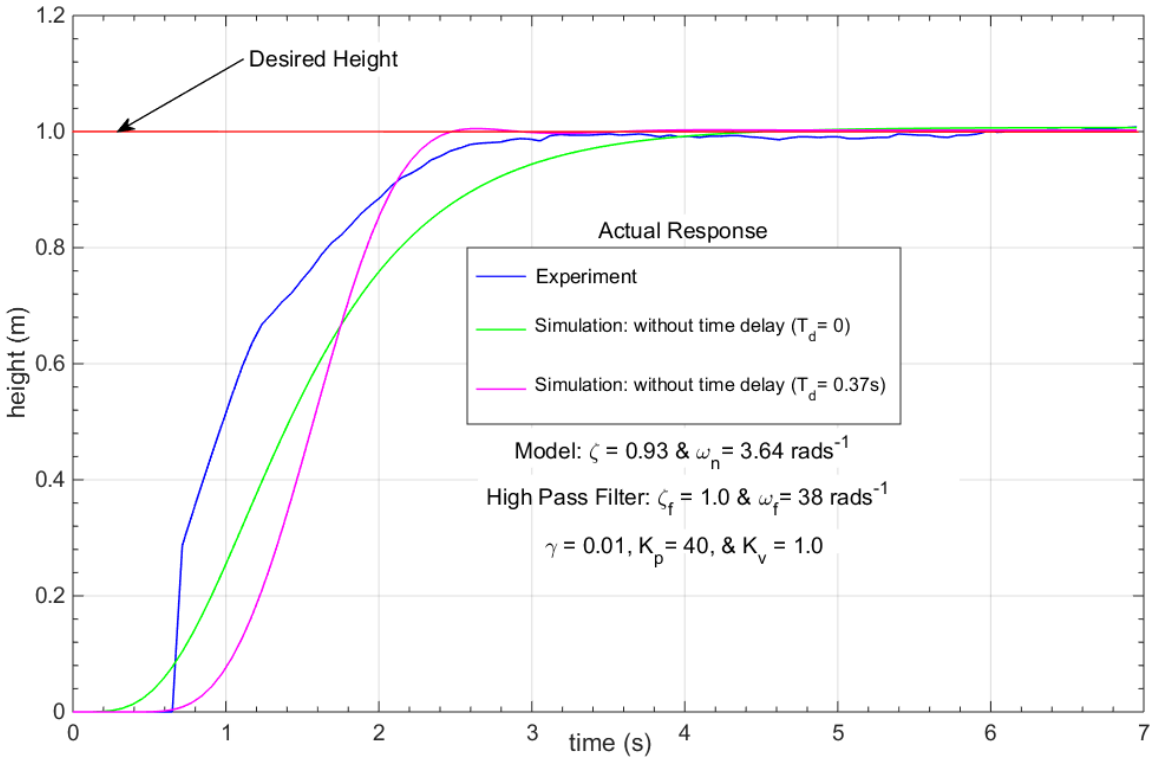


Figure 7.34. PV-MRAC controller responses: black-box approach model.

Table 7.15

PV-MRAC Controller Altitude Responses: Black-box Approach Model

	Experiment	Simulation ($T_d = 0$)	Simulation ($T_d = 0.37$s)
M_o (%)	0.3000	0.6950	0.5161
t_p (s)	3.4650	6.9555	2.6650
t_s (s)	2.0475	3.5963	2.3244
ESS (m)	0.0003	0.0069	0.0026
RMS	0.7306	0.3080	0.3956

Moreover, Figure 7.35 shows experimental responses of the PV-MRAC controller as against using only the PV controller. The PV controller on its own never stabilized the system; the response oscillates. Thus, it can be concluded that increasing γ makes the PV controller becomes

more dominant in the control of the system, while decreasing γ makes the MRAC the dominant controller.

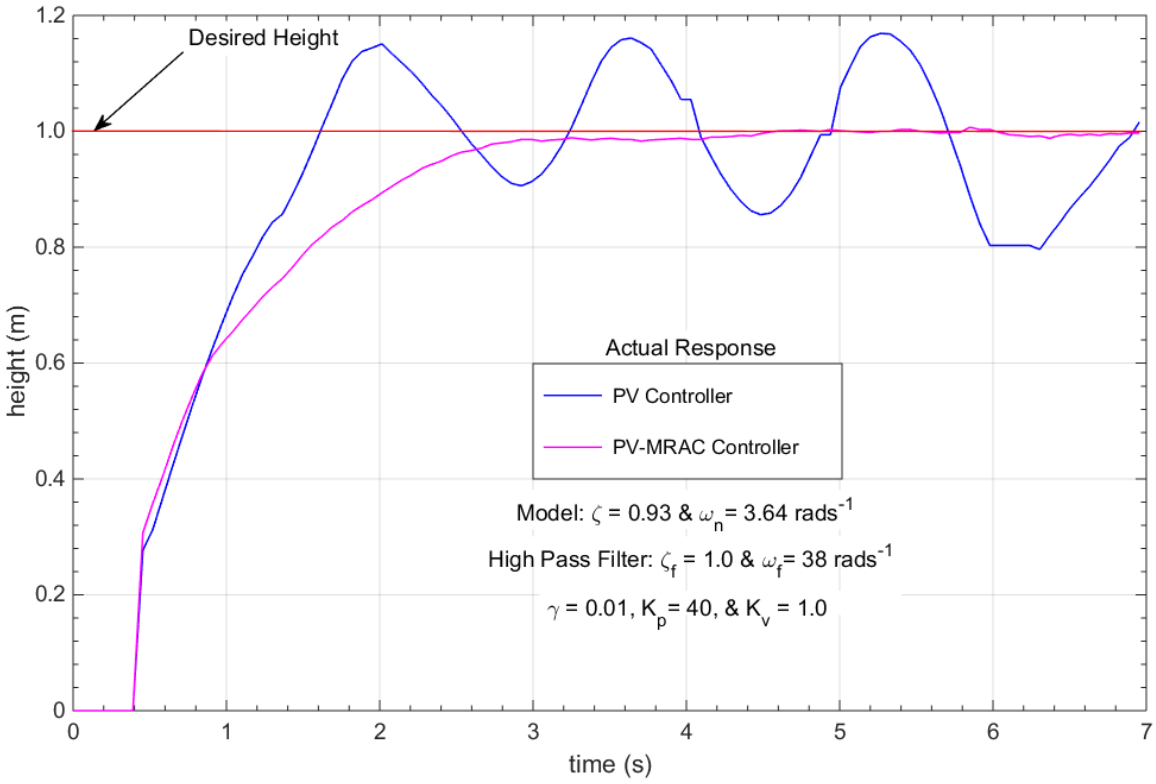


Figure 7.35. Experimented PV-MRAC controller versus only PV controller responses.

7.6 Effects of System Uncertainty

7.6.1 Real stability radius computation.

7.6.1.1 First-order model. Analytically, as stated in *Section 5.5.1.1*, $r_{\Re} = 0.8755$, $\|\Delta\| = \|\Delta_B\| = 0.8755$, and $\Delta_{K_g} = \pm 0.6683 rads^{-1}V^{-1}$ were obtained. Also, from the statistical-experimental results, see *Section 3.2.7.1*, $\Delta_B = \pm 0.6461$ was obtained, thus $\Delta_{K_g} = \pm 0.4932 rads^{-1}V^{-1}$ and $\|\Delta_B\| = 0.6461$. Therefore, comparing the results, $\|\Delta_B\| = 0.6461 < 0.8755$ or $\left|\Delta_{K_g}\right| = 0.4932 < 0.6683$, it can be concluded that the stability of the closed-loop system in (3.123) or (3.124) is guaranteed if it is subjected to the random uncertainty in equation (3.117). Figure 7.36 shows altitude responses from simulation by varying Δ_{k_g}. It can be seen that

as $\Delta_{k_g} \cong > 0.6683$, the response becomes faster, with increased oscillations and the system become unstable. On the other hand, as $\Delta_{k_g} \cong < -0.6683$, the response becomes slower, with no oscillations and the system becomes unstable.

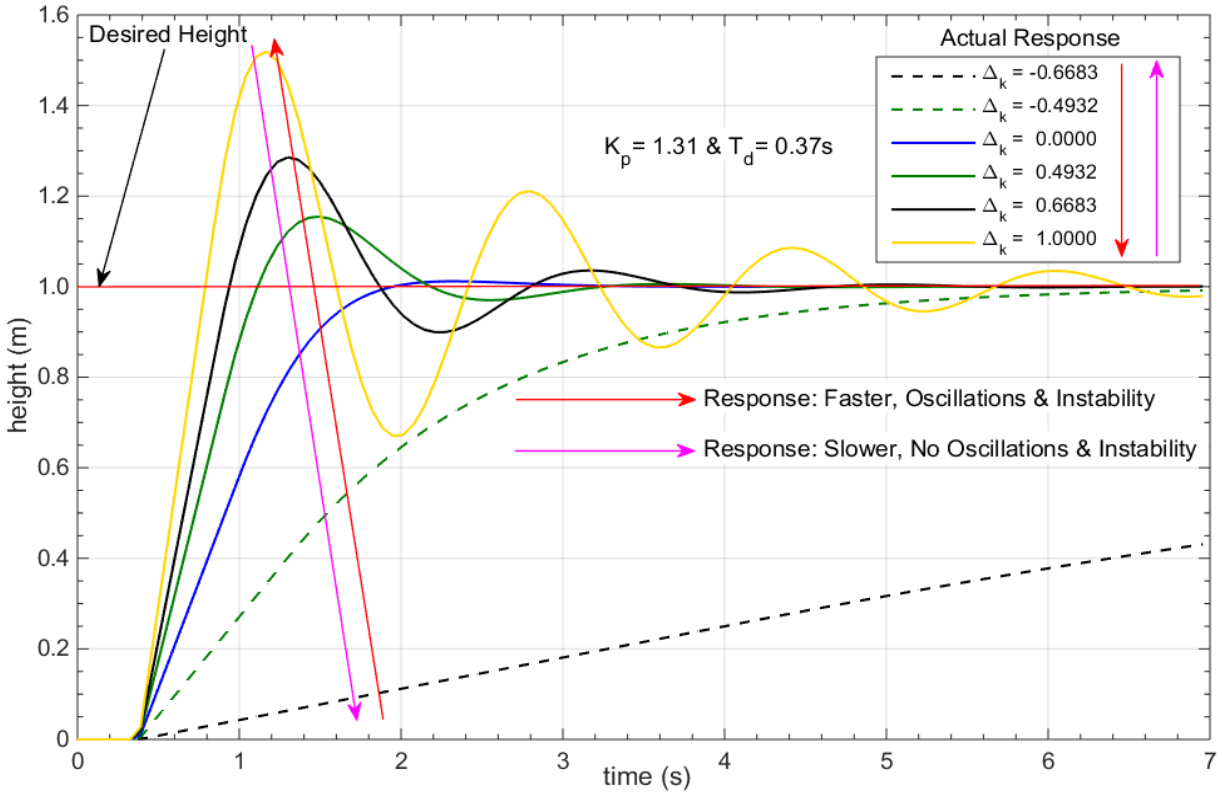


Figure 7.36. P control system altitude responses: uncertain first-order model.

7.6.1.2 Second-order model: reduced-form. Analytically, the smallest destabilizing perturbation matrix minimum norm, $\|\Delta\| = \|[\Delta_A,\ \Delta_B]\| = 0.5494$, see *Section 5.5.1.2*. Table 7.16 shows the comparison of this value to the sample of possible values from experiment, see Tables 3.3 and 5.1. Figure 7.37 shows samples of altitude responses using different values of Δ_{k_g} and Δ_τ. It can be seen that stability of the control system for Cases 1 and 2 has been confirmed. It can also be observed that though the stability of Case 3 was not guaranteed, the control system is actually 'suitably' stable. Furthermore, Figure 7.37 shows a response for an unstable control system using $\Delta_{k_g} = 1.0000 rads^{-1}V^{-1}$ and $\Delta_\tau = 0.5000s$.

Table 7.16

Comparison of Analytical and Experimental Values of $\|\Delta\|$

	Case 1	Case 2	Case 3
	$\Delta_{K_g} = 0.0100$ & $\Delta_\tau = 0.0000$	$\Delta_{K_g} = 0.0100$ & $\Delta_\tau = -0.0010$	$\Delta_{K_g} = -0.2000$ & $\Delta_\tau = -0.0100$
Analytical, $\|\Delta\|$	0.5494	0.5494	0.5494
Experimental, $\|\Delta\|$	0.1010	0.1847	1.5272
Comparison	$0.1010 < 0.5494$	$0.1847 < 0.5494$	$1.5272 > 0.5494$
Conclusion	Stability is guaranteed	Stability is guaranteed	Stability is not guaranteed

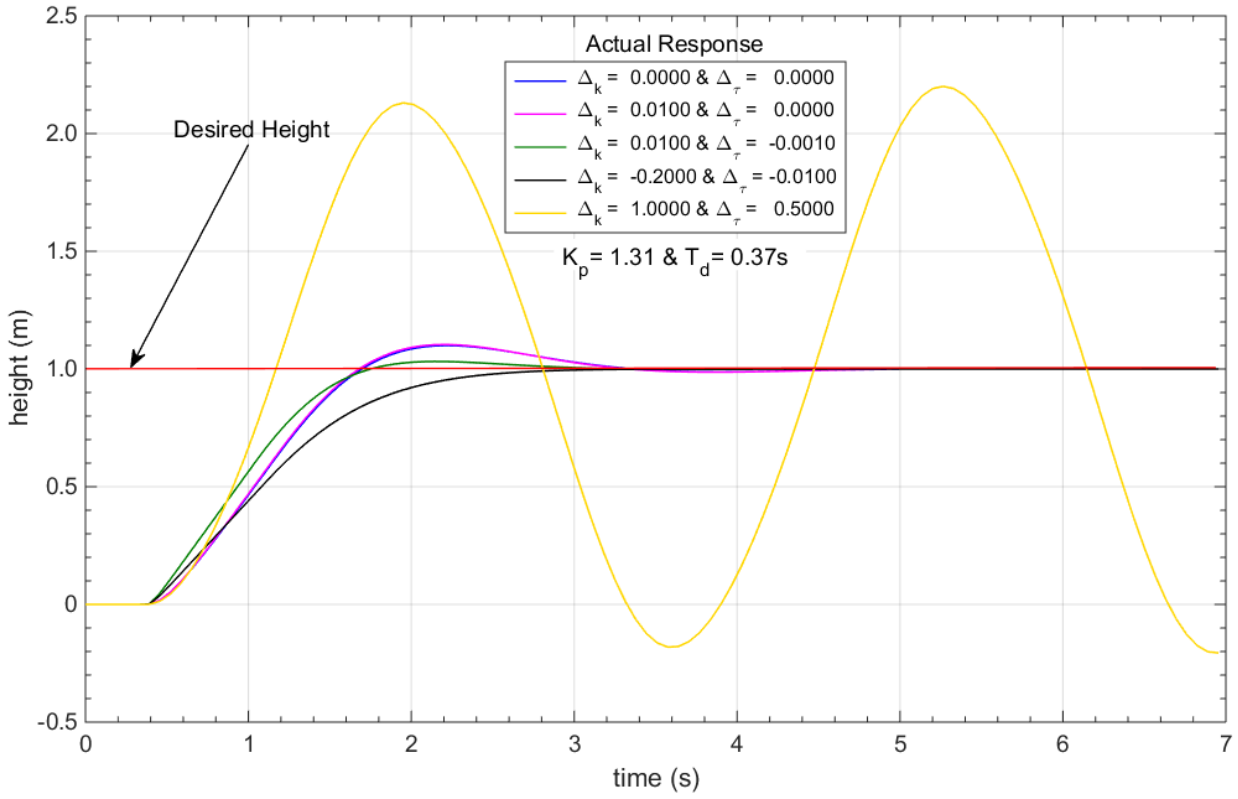


Figure 7.37. P control system altitude responses: uncertain second-order model.

7.7 Autonomous Waypoints Tracking and Effects of Disturbance Rejection

7.7.1 Autonomous waypoints tracking. The altitude responses for the autonomously multiple waypoints tracking example using the PV and PV-MRAC control systems are shown Figures 7.38 to 7.40, refer to *Section 5.6.1*. Both controllers successfully achieved the task.

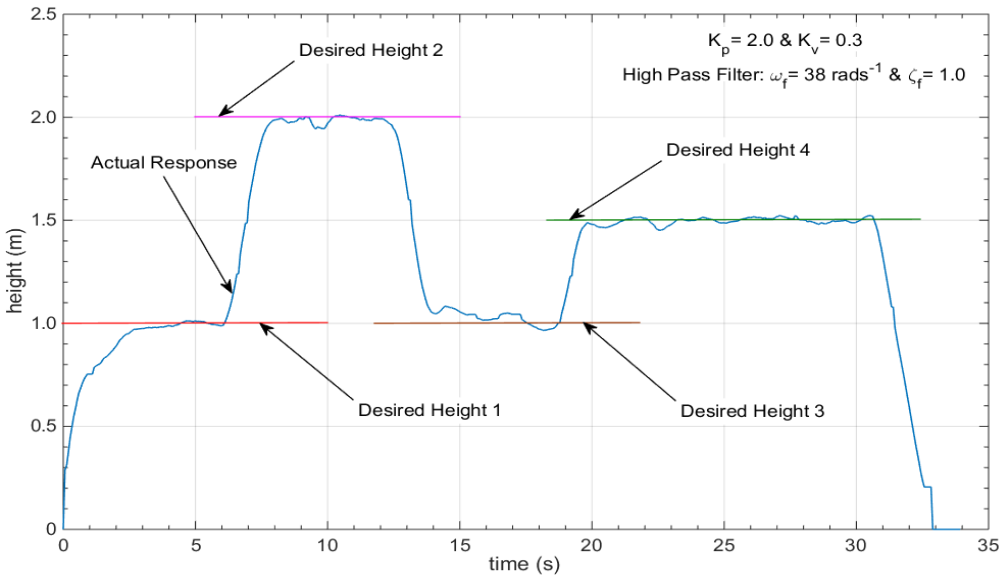


Figure 7.38. Experimented PV controller altitude response: waypoints tracking.

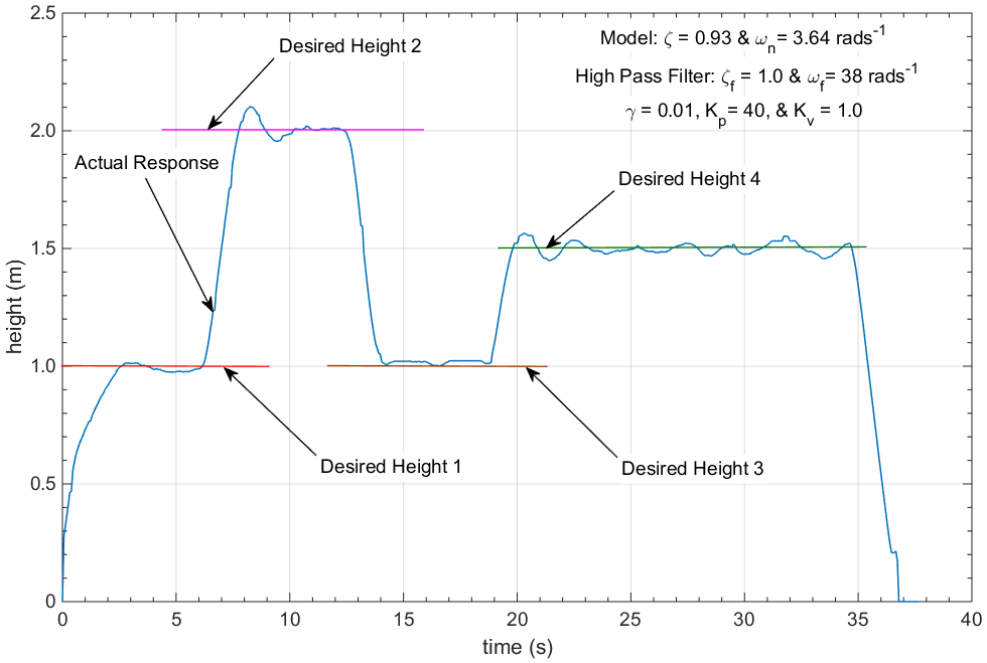


Figure 7.39. Experimented PV-MRAC controller altitude response: waypoints tracking.

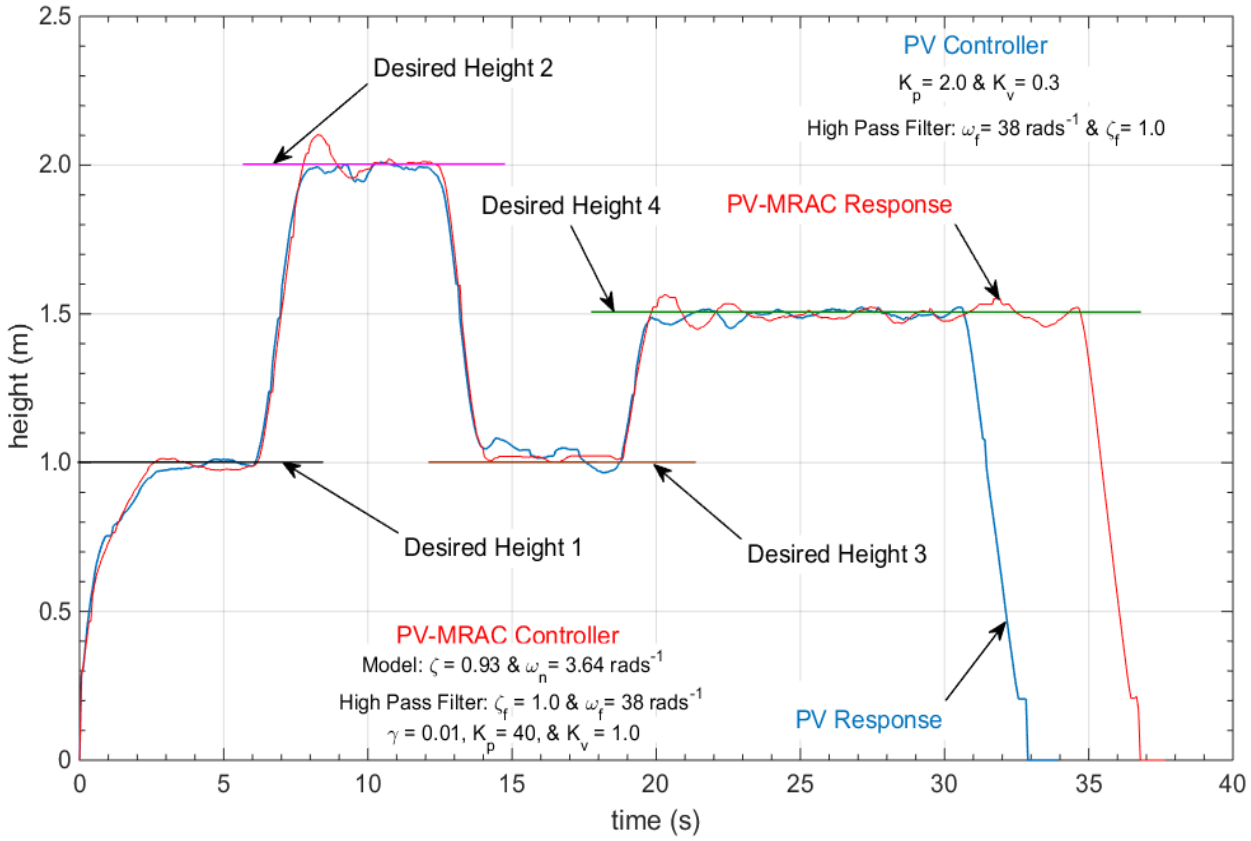


Figure 7.40. Experimented PV and PV-MRAC controllers responses: waypoints tracking.

7.7.2 Effects of disturbance rejection: quadrotor payload. Figures 7.41 to 7.48 show the responses for the three cases considered in introducing disturbances to the drone, see *Section 5.6.2*. In general, the results confirm that the PV-MRAC offers several benefits over the fixed-gain approach, PV controller. The PV-MRAC was found to offer enhanced robustness to the change in mass uncertainty and the oscillating disturbances.

7.7.2.1 Case 1. Figure 7.41 shows the PV-MRAC altitude responses by varying the mass attached at the top of the drone. It can be seen that with $130g$ attached, the drone's oscillations about the desired height became obvious. It was observed that beyond this value the oscillations were larger, which destabilizes the drone (e.g., see the $135g$ attached mass response). The drone's stability bounds reached at $130g$.

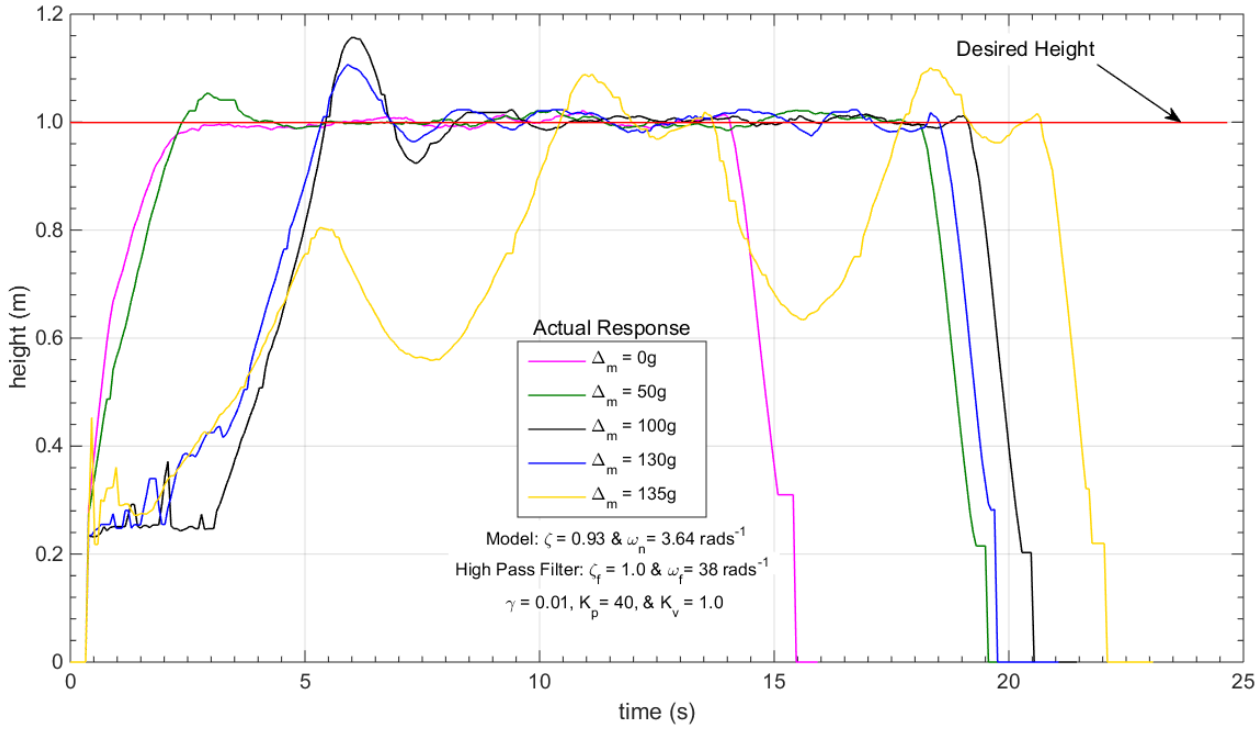


Figure 7.41. Experimented PV-MRAC responses: varying attached mass at the top.

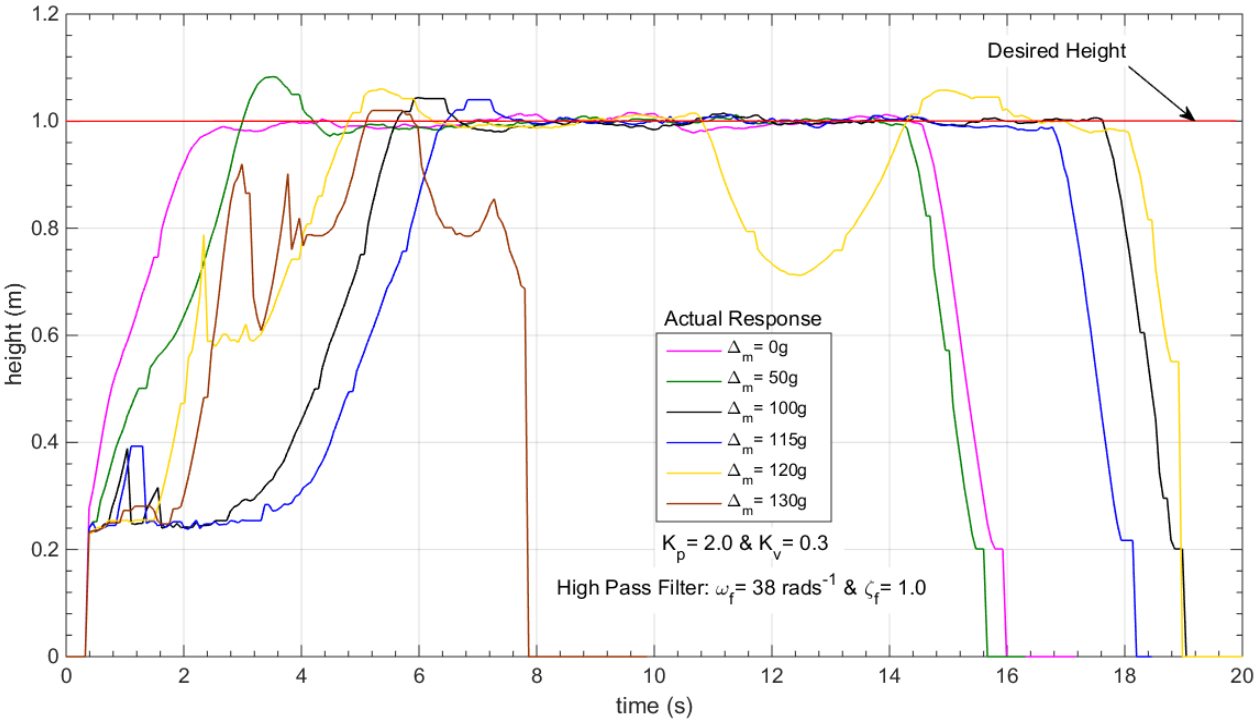


Figure 7.42. Experimented PV responses: varying attached mass at the top.

Figure 7.42 shows the PV altitude responses by varying the mass attached at the top of the drone. It can be seen that with $130g$ attached, the drone was destabilized. It was observed that beyond $125g$ of attached mass the oscillations were larger, which destabilizes the drone. The drone's stability bounds reached at $120g$.

Figure 7.43 shows the altitude responses for the PV and PV-MRAC controllers for the case of attaching $130g$ mass at the top of the drone. The difference in performance of the controllers was obvious, the adaptive controller allowed safe operation and landing, while the PV controller failed to prevent instability and sometimes resulted in a crash landing.

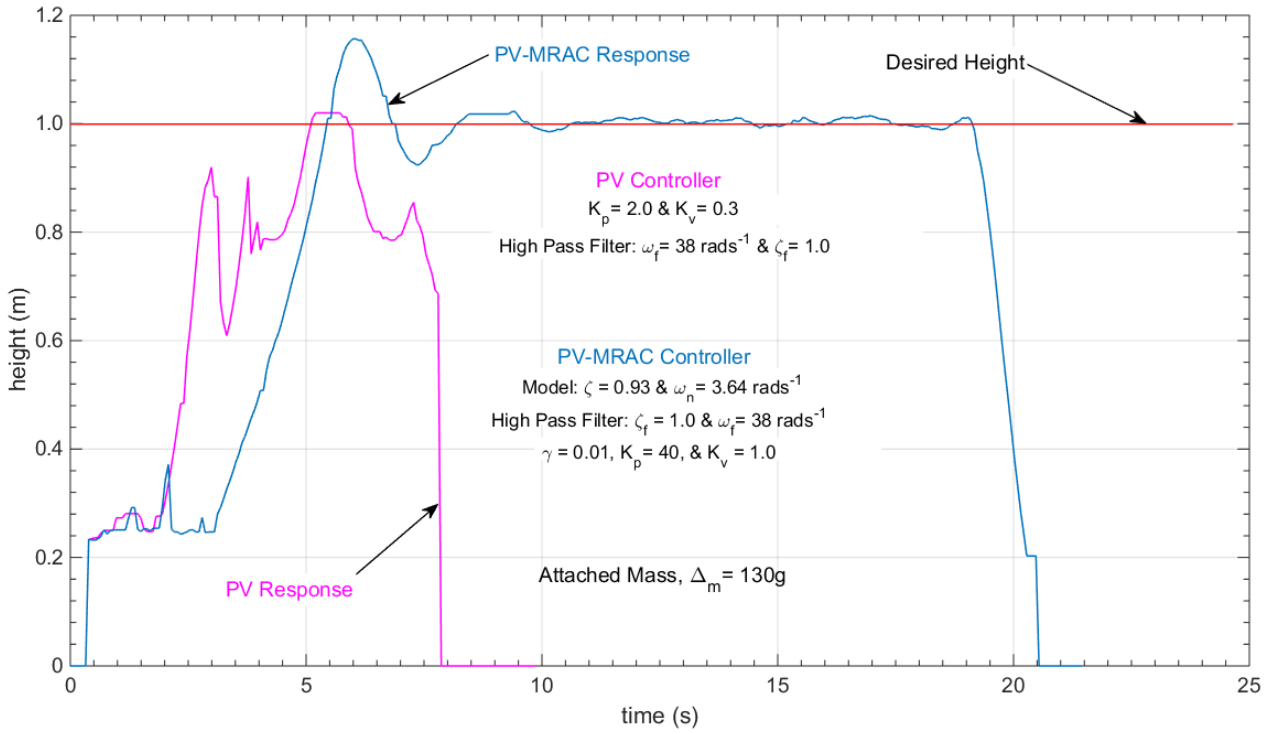


Figure 7.43. Experimented PV and PV-MRAC responses: $130g$ attached mass at the top.

7.7.2.2 Case 2. The case of attaching $100g$ hanging mass, which introduced oscillating disturbances, both controllers achieved the task of rising to the desired height, however, the PV-MRAC controller still performed better, see Figure 7.44. The PV-MRAC controller response has smaller oscillations about the desired height value compared to the PV controller response.

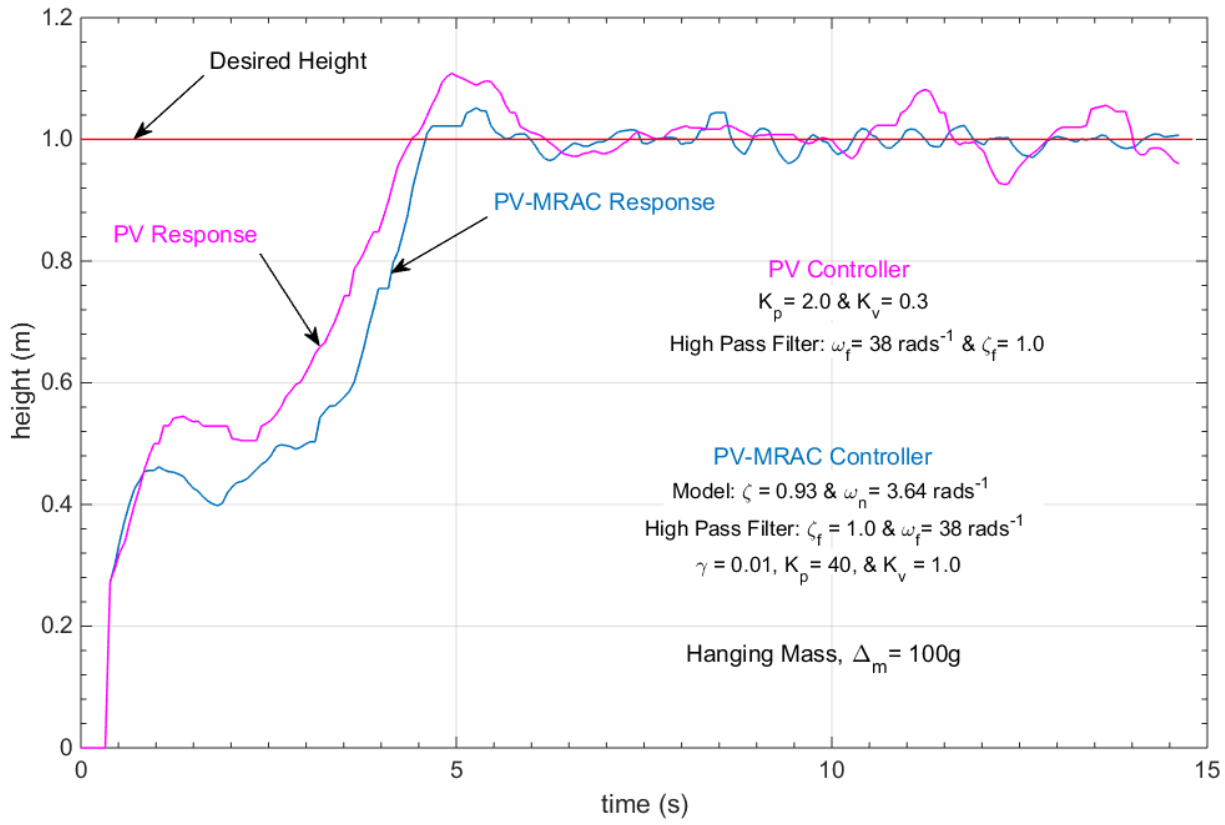


Figure 7.44. Experimented PV and PV-MRAC responses: $100g$ hanging mass.

7.7.2.3 Case 3. Figures 7.45 and 7.46 show the responses for the PV and PV-MRAC controllers, respectively, for the case when instantaneous masses of $100g$ and $135g$ were attached at the top of the drone. For the $100g$ mass, both controllers performed well. After the introduction of the mass, the drone dropped from its desired height of $1m$ to about $0.55m$, and then moved back to the desired height with a little bit of oscillations in the responses. However, when the $135g$ mass was introduced, the PV controller failed to prevent instability and resulted in a crash landing, but the adaptive controller maintained the stability and allowed safe operation and landing. Figures 7.47 and 7.48 show the responses for the PV and PV-MRAC controllers, respectively, for the case when varying instantaneous masses were attached at the top of the drone. The drone's stability bounds reached at $125g$ and $135g$ using the PV and PV-MRAC controllers, respectively.

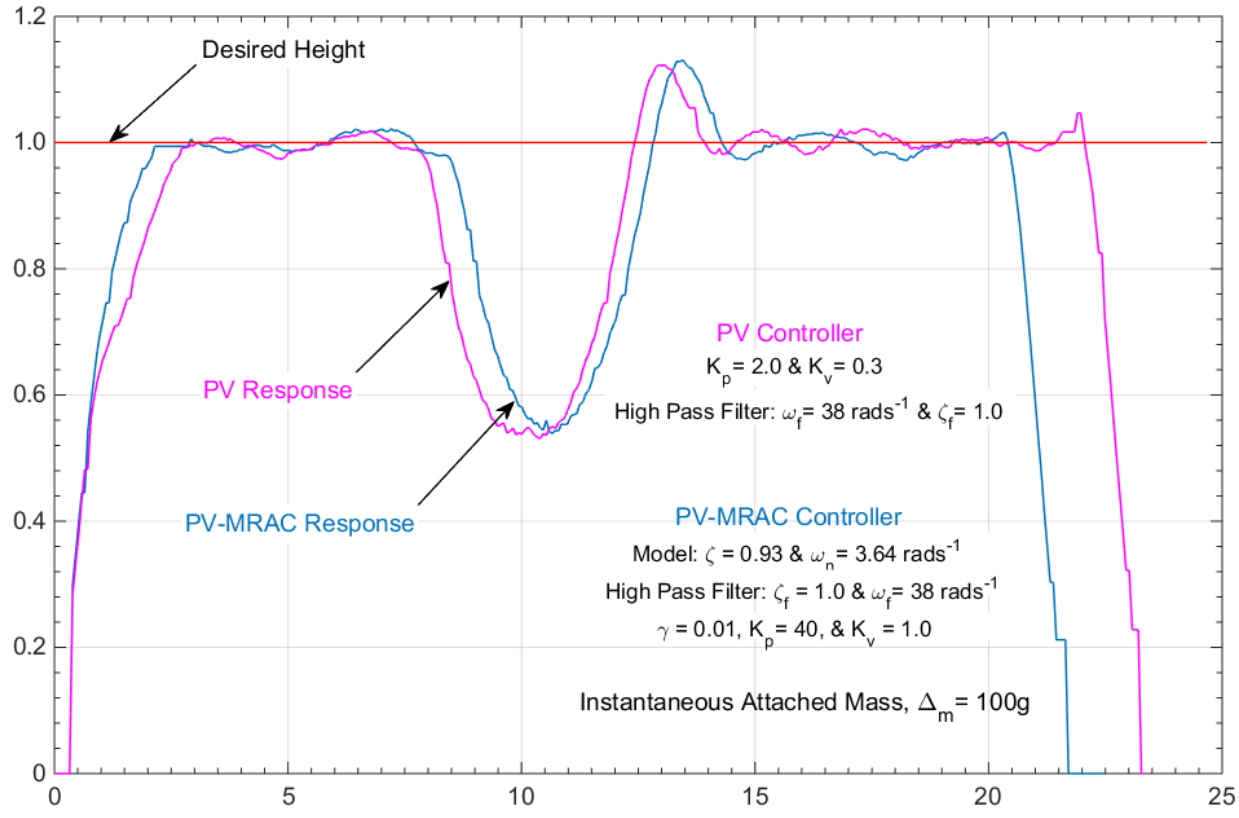


Figure 7.45. Experimented PV and PV-MRAC responses: $100g$ instantaneous mass.

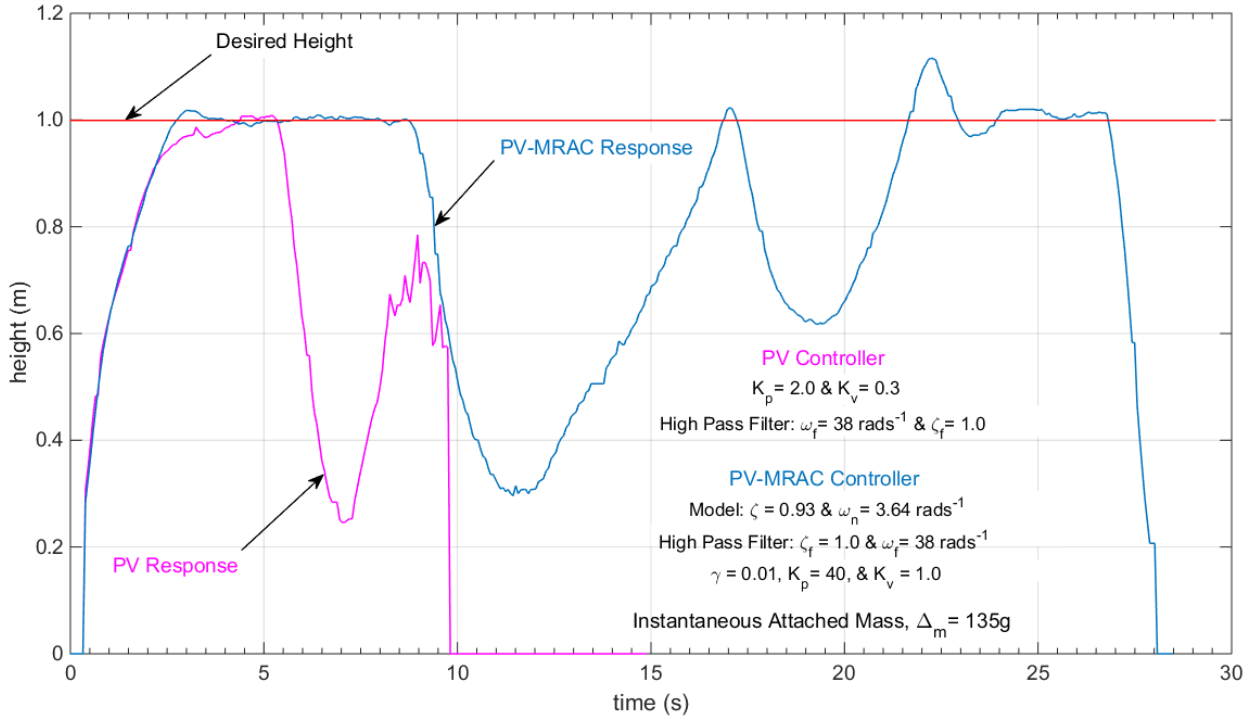


Figure 7.46. Experimented PV and PV-MRAC responses: $135g$ instantaneous mass.

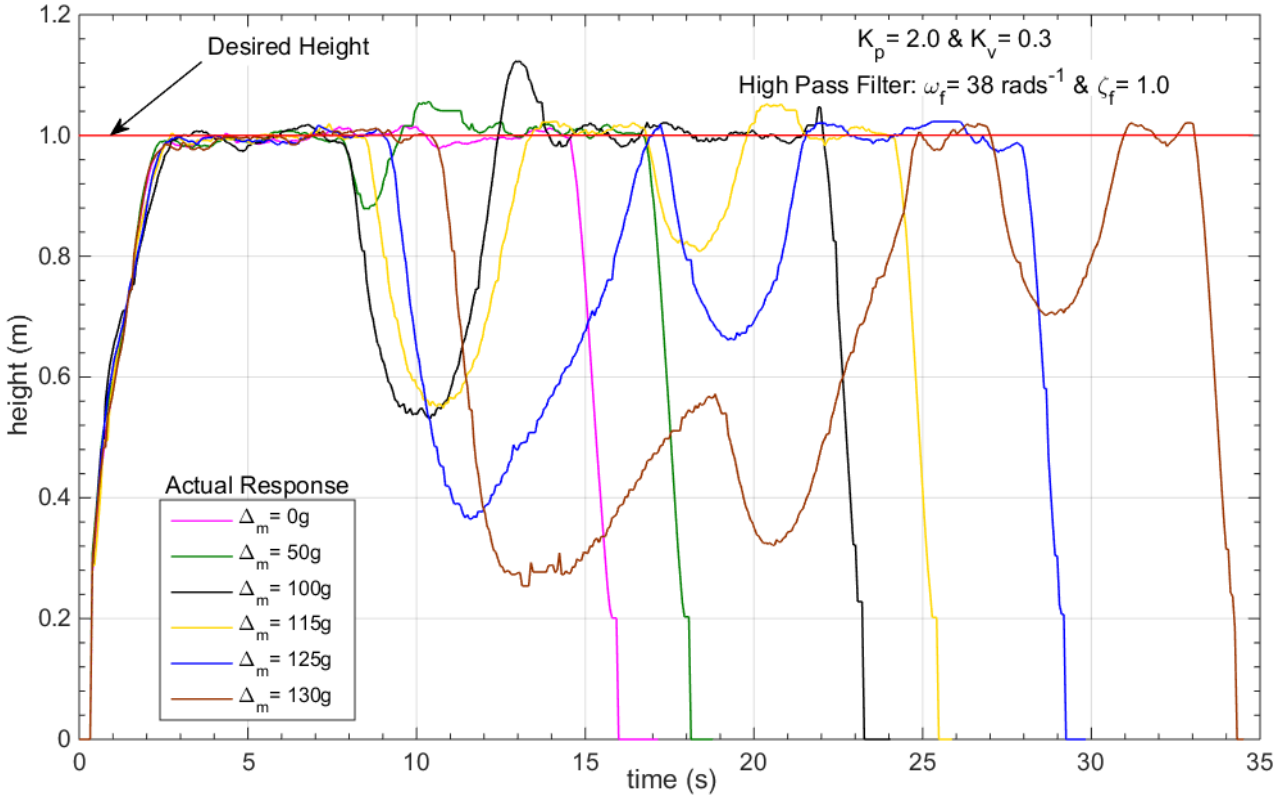


Figure 7.47. Experimented PV responses: varying instantaneous mass at the top.

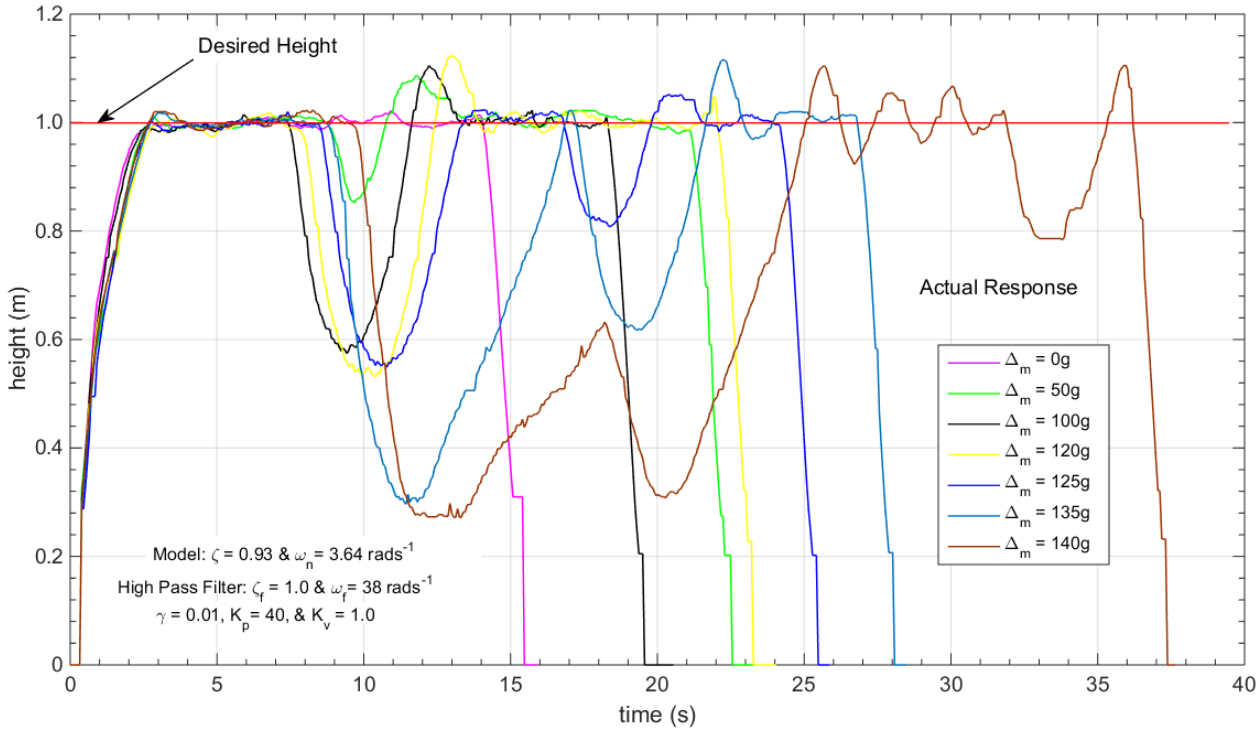


Figure 7.48. Experimented PV-MRAC responses: varying instantaneous mass at the top.

CHAPTER 8

Conclusions

This research work started with developing an effective autonomous navigation system for the DDWMR, Dr Robot X80SV, and this was achieved using logic control. The developed navigation system was validated via simulating using a MATLAB robot simulator and implementation on the X80SV platform using a C# program. The algorithm successfully makes the robot move to a pose while avoiding obstacles along the way. For example, for a desired pose of $(2m, 0m, 0^o)$, the final pose experimentally obtained was $(2.0197m, 0.0266m, -0.5500^o)$. The average stabilization time was about $44s$ and the robot linear constant speed was $0.5ms^{-1}$. Even though the final steady-state values obtained from the experiments have small errors, the results were encouraging in terms of speed and accuracy. According to Brockett's conditions [77], PID feedback laws cannot be used to obtain error-free stabilization.

It was demonstrated that the nonlinear equations of motion can be used to control the WMR; however, some challenges were encountered. The main challenge was in sending signals to the DC motors and receiving measured data using the developed MATLAB GUI program in real time. The research continued to develop autonomous position (x, y, and z) and attitude (roll, pitch, and yaw) control for quadrotor, and then implementing the controllers on the Parrot AR.Drone 2.0. At the next stage of the research, PD-feedback controllers for position and attitude control of quadrotor UAVs were developed. MATLAB/Simulink models were used to design and validate the control algorithms, and the simulation results were presented. A robust and interactive MATLAB GUI interface developed to implement the different control algorithm simulation models was also presented. Considering the final steady-state values and transient performance obtained from the simulations the results were also quite encouraging.

In implementing these PD-feedback controllers developed on the drone, several challenges and difficulties were encountered. The main difficulty arose in using the MATLAB S-function to implement the nonlinear dynamics and kinematics of quadrotors. The problem was in C/C^{++} code generation for real-time application. In order to address these problems, the white-box approach was used to obtain a linearized altitude model from the dynamics and kinematics of the quadrotor. Then P, pole placement or PV, LQR, and MRAC controllers were designed and validated using MATLAB/Simulink. Nonlinear effects of control input saturation and time delay in the control systems were studied. The P and PV controllers successfully improved the system performance in the terms of time-domain specification.

The difficulties with the MRAC implementation was addressed by designing a new controller based on the second-order LTI ARX altitude model (black-box approach) using MATLAB system identification toolbox. Finally, after some initial challenges of finding the right tuning parameters and problems of solvers, a PV-MRAC controller was designed and implemented on the drone. The MATLAB GUI and Simulink programs developed for the control of the drone's altitude motion have also been presented.

The research also demonstrated the use of analytical and experimental methods to estimate the time delay in a quadrotor UAV, AR.Drone 2.0 control system. A first-order model was used for the analytical determination of the time delay and for obtaining the simulation altitude responses. The time delay was estimated as $0.374s$ and $0.368s$ using analytical and experimental methods, respectively. Therefore, in the drone's control system the time delay was approximately $0.37s$. P-feedback control system (retarded type) was used in the estimation of the time delay. An appropriate P controller, $K_p = 1.31$, was used to minimize the effect of the applied control signal

saturation on the system's response, especially as K_p increases. MATLAB/Simulink was used for the simulations, experiments, and analytical solutions of the DDEs.

Then, an improved system performance was obtained by incorporating the estimated time delay in the design of the PV control system (neutral type). The results show that the PV controller performed better than the P controller, especially with gains of $K_v = 0.3, 0.5,$ and 0.7 at $K_p = 2.0$. Also, improved tuning parameters were obtained for the PV-MRAC control system after the time delay has been taken into account. An example of the appropriate tuned parameters for the PV-MRAC control system are $\gamma = 0.01, K_v = 1.0,$ and $K_p = 40$ for the controller, and $\omega_n = 3.64 rads^{-1}$ and $\xi = 0.93$ for the reference model. The suitable high pass filter which was designed for the V controller has a damping ratio of 1.0, natural frequency of $38 rads^{-1}$, and with a cutoff frequency of $0.90 Hz$.

Furthermore, the stability of a parametric perturbed LTI time-delayed P control system (retarded type) was studied. This was done by analytically calculating the stability radius of the system. The destabilizing, worst, perturbation matrix minimum norms were compared from the analytical and experimental results to predict the robust stability of the system subjected to parametric disturbances. Then, simulation of the control system was conducted to confirm the stability. This robust control design and uncertainty analysis were conducted for first-order and second-order systems.

Moreover, the designed PV and PV-MRAC control systems were used to autonomously track multiple specified waypoints, and both controllers successfully achieved the task. Also, the robustness of the PV-MRAC controller was tested against a baseline, PV controller, using the payload capability of the drone and the introduction of oscillating disturbances to the system. It was shown that the PV-MRAC offers several benefits over the fixed-gain approach of the PV

controller. The PV-MRAC was found to offer enhanced robustness to the parametric (uncertainty in mass) and the oscillating disturbances. The difference in performance of the controllers was obvious. The adaptive controller allowed safe operation and landing while the PV controller failed to prevent instability and sometimes resulted in a crash landing. The drone's stability bounds reached at $120g$ and $130g$ for added attached mass-inertia using the PV and PV-MRAC controllers, respectively. In the case of the instantaneous introduction of the mass-inertia, the drone's stability bounds reached at $125g$ and $135g$ using the PV and PV-MRAC controllers, respectively.

In future, the issue of sending signals to the X80SV DC motors and receiving measured data using the MATLAB program in real time needs to be addressed. Also, the estimated time delay can be incorporated in designing effective and robust controllers for the drone's attitude and position (x and y) motions. Furthermore, the MRAC can be combined with gain scheduling in controlling the drone's motions. Moreover, the presented time-delay estimating methods can be extended to general systems of DDEs (higher than first order), and be applied to delay problems in network systems and fault detection of actuators. Additionally, a robust control strategy that optimizes parameters such as length of path or journey time can be studied further, and real-time vision-based object detection and recognition can be incorporated.

References

[1] "Humanoid mobile robot," *Retrieved November 14, 2014, from aldebaran.com/en.*

[2] "2-four rotors quadrotor," *Retrieved November 4, 2014, from wilddrones.com/.*

[3] "Personal transportation mobile robot," *Retrieved November 12, 2014, from segway.com/.*

[4] "Quadrotor mobile robot," *Retrieved November 3, 2014, from robots.net/article/2992.html.*

[5] "Car-like mobile robot," *Retrieved November 8, 2014, from neurotechnology.com/robotics.html.*

[6] "Harvey robot: harvest automation," *Retrieved November 2, 2014, from harvestai.com/.*

[7] "Two wheeled platform mobile robot," *Retrieved November 12, 2014, from ardrone2.parrot.com/.*

[8] "3-four rotors quadrotor," *Retrieved November 4, 2014, from dailymail.co.uk/news/article-2395933/.*

[9] "NASA curiosity: mars exploration," *Retrieved November 1, 2014, from nasa.gov/mission_pages/.*

[10] "Six wheeled platform mobile robot," *Retrieved November 12, 2014, from dai.ed.ac.uk/groups/mrg/robots.html.*

[11] "Commercial drones," *Retrieved March 4, 2015, from dronesim.com/faa-investigates-tornado-photos-from-drone.*

[12] "Commercial drones," *Retrieved March 4, 2015, from upriser.com/posts/us-moves-toward-opening-skies-for-commercial-drones.*

[13] "Snake mobile robot," *Retrieved November 2, 2014, from industrytap.com/surgical-snake-robots-accessing-organs-and-tissues-deep-in-the-body/2051.*

[14] "Cheetah mobile robot," *Retrieved November 12, 2014, from industrytap.com/surgical-snake-robots-accessing-organs-and-tissues-deep-in-the-body/2051.*

[15] S. Vijay, "Autonomous underwater vehicle," *Department of Mechanical ENGG., P.E.S.C.E., MANDYA.*

[16] "Quadcopter," *Retrieved February 23, 2015, from en.wikipedia.org/wiki/Quadcopter.*

[17] "World robotics 2014 service robots," *Retrieved March 2, 2015, from ifr.org/service-robots/statistics/.*

[18] M. Meckstroth, "Mobile robotics: moving robots forward," *Retrieved February 23, 2015, from rtcmagazine.com/articles/view/101197.*

[19] C. Anderson, "Future vision of the world," *Retrieved November 4, 2014, from gigaom.com/2013/09/26/3d-robotics-ceo-chris-anderson-on-the-future-of-drones/.*

[20] J. Simpson, C. L. Jacobsen, and M. C. Jadud, "Mobile robot control," in *Communicating Process Architectures 2006: WoTUG-29: Proceedings of the 29th WoTUG Technical Meeting, 17-20 September 2006, Napier University, Edinburgh, Scotland*, 2006, p. 225.

[21] "Adaptive control of Very Flexible Aircaft," *Retrieved March 16, 2015, from mit.edu/~tgibson/research.html.*

[22] "Commercial drones," *Retrieved March 4, 2015, from computerworld.com/article/2599405/emerging-technology.*

[23] N. K. Sinha, M. M. Gupta, and P. N. Nikiforuk, "Recent advances in adaptive control," *Cybernetics and System,* vol. 6, pp. 79-100, 1976.

[24] C. R. Anderson, H. L. Chang, and J. S. Gibson, "Adaptive control of vortex dynamics," in *Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on*, 1993, pp. 1909-1910.

[25] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Adaptive control of quadrotor UAVs: a design trade study with flight evaluations," *Control Systems Technology, IEEE Transactions on,* vol. 21, pp. 1400-1406, 2013.

[26] Z. S. Alavi, M. B. Menhaj, and H. Eliasi, "Model reference adaptive control of a nuclear reactor," in *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, 2009, pp. 735-740.

[27] "Adaptive control for a Biomass plant," *Retrieved March 16, 2015, from adexcop.com/studies-24.*

[28] "Robot positioning with metrology accuracy," *Retrieved March 16, 2015, from nikonmetrology.com/en_US.*

[29] "Parrot AR.Drone 2.0," *Retrieved October 9, 2014, from mentalmunition.com/2012/07/maker-of-ar-drone-invests-in-gis-is.html.*

[30] H. Mo, Q. Tang, and L. Meng, "Behavior-based fuzzy control for mobile robot navigation," *Mathematical Problems in Engineering,* vol. 2013, 2013.

[31] J. J. Park, C. Johnson, and B. Kuipers, "Robot navigation with model predictive equilibrium point control," in *IROS*, 2012, pp. 4945-4952.

[32] K. U. K. Lee, Y. H. Yun, W. Chang, J. B. Park, and Y. H. Choi, "Modeling and altitude control of quad-rotor UAV," in *Control, Automation and Systems (ICCAS), 2011 11th International Conference on*, 2011, pp. 1897-1902.

[33] M. H. Tanveer, S. F. Ahmed, D. Hazry, F. A. Warsi, and M. K. Joyo, "Stabilized controller design for attitude and altitude controlling of quad-rotor under disturbance and noisy conditions," *American Journal of Applied Sciences,* vol. 10, p. 819, 2013.

[34] W. Zhou, K. Yin, R. Wang, and Y. E. Wang, "Design of attitude control system for UAV based on feedback linearization and adaptive control," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[35] A. Ailon and S. Arogeti, "Study on the effects of time-delays on quadrotor-type helicopter dynamics," in *Control and Automation (MED), 2014 22nd Mediterranean Conference of*, 2014, pp. 305-310.

[36] Q. Wang and R. F. Stengel, "Robust control of nonlinear systems with parametric uncertainty," *Automatica*, vol. 38, pp. 1591-1599, 2002.

[37] G. Hu and E. J. Davison, "Real stability radii of linear time-invariant time-delay systems," *Systems & Control Letters*, vol. 50, pp. 209-219, 2003.

[38] A. De Luca, G. Oriolo, and M. Vendittelli, "Control of wheeled mobile robots: an experimental overview," in *Ramsete*, ed: Springer, 2001, pp. 181-226.

[39] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB* vol. 73: Springer Science & Business Media, 2011.

[40] "Mobile robot navigation," *Retrieved March 8, 2015, from en.wikipedia.org/wiki/Mobile_robot_navigation.*

[41] "Types of mobile robots," *Retrieved November 6, 2014, from en.wikipedia.org/wiki/Mobile_robot.*

[42] K. Ogata, *Modern control engineering*: Prentice-Hall, 2002.

[43] K. Ogata, *Matlab for control engineers*.

[44] "PID controller," *Retrieved December 24, 2014, from en.wikipedia.org/wiki/PID_controller.*

[45] I. D. Landau, R. Lozano, M. M'Saad, and A. Karimi, *Adaptive control: algorithms, analysis and applications*: Springer Science & Business Media, 2011.

[46] "Adaptive control," *Retrieved March 5, 2015, from en.wikipedia.org/wiki/Adaptive_control.*

[47] "MIT rule MRAC," *Retrieved November 2, 2014, from pages.drexel.edu/~kws23/tutorials/MRAC/theory/theory.html.*

[48] "Robust control," *Retrieved March 5, 2015, from en.wikipedia.org/wiki/Robust_control.*

[49] "Root locus plot," *Retrieved March 2, 2015, from mathworks.com/help/control/ref/rlocus.html.*

[50] "SISO design tool," *Retrieved March 2, 2015, from mathworks.com/help/control/getstart/siso-design-tool.html?refresh=true.*

[51] "System identifcation," *Retrieved March 3, 2015, from en.wikipedia.org/wiki/System_identification.*

[52] "System identification toolbox," *Retrieved March 2, 2015, from mathworks.com/help/ident/index.html.*

[53] "Transient response," *Retrieved January 12, 2015, from en.wikipedia.org/wiki/Transient_response.*

[54] W. J. Palm, *System dynamics*, 2nd ed.: McGraw-Hill Higher Education, 2010.

[55] P. H. A. Ngoc and T. Naito, "Stability radius of linear parameter-varying systems and applications," in *Decision and Control, 2006 45th IEEE Conference on*, 2006, pp. 5736-5741.

[56] K. Mahdinejad and M. Z. Seghaleh, "Implementation of time delay estimation using different weighted generalized cross correlation in room acoustic environments."

[57] X. M. Ren, A. B. Rad, P. T. Chan, and W. L. Lo, "Online identification of continuous-time systems with unknown time delay," *IEEE transactions on automatic control,* vol. 50, pp. 1418-1422, 2005.

[58] S. Yi, W. Choi, and T. Abu-Lebdeh, "Time-delay estimation using the characteristic roots of delay differential equations," *American Journal of Applied Sciences,* vol. 9, p. 955, 2012.

[59] L. Belkoura, J. P. Richard, and M. Fliess, "Parameters estimation of systems with delayed and structured entries," *Automatica,* vol. 45, pp. 1117-1125, 2009.

[60] J. P. Richard, "Time-delay systems: an overview of some recent advances and open problems," *automatica,* vol. 39, pp. 1667-1694, 2003.

[61] B. Svante, "A survey and comparison of time-delay estimation methods in linear systems," *Retrieved January 14, 2015, from control.isy.liu.se/research/,* 2003.

[62] P. Bliman, "Lyapunov equation for the stability of linear delay systems of retarded and neutral type," *Automatic Control, IEEE Transactions on,* vol. 47, pp. 327-335, 2002.

[63] "Lambert W function," *Retrieved February 10, 2015, from en.wikipedia.org/wiki/Lambert_W_function.*

[64] S. Yi, P. W. Nelson, and A. G. Ulsoy, *Time-delay systems: analysis and control using the Lambert W function*: World Scientific, 2010.

[65] W. S. Levine, *Control system fundamentals*: CRC press, 1999.

[66] R. S. Figliola and D. Beasley, *Theory and design for mechanical measurements*: John Wiley & Sons, 2015.

[67] "Stability radius," *Retrieved March 1, 2015, from en.wikipedia.org/wiki/Stability_radius.*

[68] D. Hinrichsen and A. J. Pritchard, "Stability radius for structured perturbations and the algebraic Riccati equation," *Systems & Control Letters,* vol. 8, pp. 105-113, 1986.

[69] "Positive systems," *Retrieved March 1, 2015, from en.wikipedia.org/wiki/Positive_systems.*

[70] L. Qiu, B. Bernhardsson, A. Rantzer, E. J. Davison, P. M. Young, and J. C. Doyle, "A formula for computation of the real stability radius," *Automatica,* vol. 31, pp. 879-890, 1995.

[71] "Payload," *Retrieved March 10, 2015, from en.wikipedia.org/wiki/Payload.*

[72] P. E. I. Pounds, D. R. Bersak, and A. M. Dollar, "Stability of small-scale UAV helicopters and quadrotors with added payload mass under PID control," *Autonomous Robots,* vol. 33, pp. 129-142, 2012.

[73] M. Egerstedt, "Control of mobile robots (PowerPoint)," *Coursera,* 2013.

[74] R. W. Beard, "Quadrotor dynamics and control," *Brigham Young University,* 2008.

[75] H. Shinozaki and T. Mori, "Robust stability analysis of linear time-delay systems by Lambert W function: some extreme point results," *Automatica,* vol. 42, pp. 1791-1799, 2006.

[76] M. Becker, R. C. B. Sampaio, S. Bouabdallah, V. D. Perrot, and R. Siegwart, "In flight collision avoidance for a mini-UAV robot based on onboard sensors," *Mechatronics Lab,* 2012.

[77] R. W. Brockett, "Asymptotic stability and feedback stabilization," *Differential Geometric Control Theory*, pp. 181-208, 1983.

[78] T. Vyhlídal, "Quasi-polynomial mapping based rootfinder (software)," *Faculty of Mechanical Engineering, CTU in Prague,* 2013.

[79] L. F. Shampine, "Error estimation and control for ODEs," *Journal of Scientific Computing*, vol. 25, pp. 3-16, 2005.

[80] Mathworks, "Choosing a solver," *Retrieved February 16, 2015, from mathworks.com/help/optim/ug/choosing-a-solver.html*.